Working Draft
American National Standard

Project
T10/BSR INCITS 515

Revision 21
July 22, 2015

Information technology - SCSI Architecture Model - 5 (SAM-5)

T10 Technical Editor: George Penokie
Avago Technologies
4109 Manor View Dr NW
Rochester, MN 55901-3109
USA

Telephone: (952) 921-2495
Email: george.penokie@avagotech.com

Printed 10:55 AM Wednesday 22 July 2015

Points of Contact

<u>**International Committee for Information Technology Standards (INCITS) T10 Technical Committee**</u>

<u>**T10 Chair**</u>   <u>**T10 Vice-Chair**</u>
Ralph O. Weber   William Martin
Western Digital Technologies   Samsung Semiconductor, Inc
18484 Preston Road, Suite 102, PMB 178   7213 Marblethorpe Drive
Dallas, TX 75252   Roseville, CA 95747-5925
USA   USA

Telephone: (214) 912-1373   Telephone: (916) 765-6875
Email: Ralph.Weber@WDC.com   Email: bill.martin@ssi.samsung.com

T10 Web Site: http://www.t10.org

<u>**INCITS Secretariat**</u>
Suite 610
1101 K Street, NW
Washington, DC   20005-7031
USA

Telephone: (202) 737-8888
Web site: http://www.incits.org
Email: incits@itic.org

<u>**Information Technology Industry Council**</u>
Web site: http://www.itic.org

<u>**Purchase INCITS Standards**</u>
Web site: http://www.incits.org/standards-information/purchase-standards-or-download-dpans

American National Standard
for Information Technology

SCSI Architecture Model - 5 (SAM-5)

Secretariat
Information Technology Industry Council

Approved mm.dd.yy

American National Standards Institute, Inc.

ABSTRACT

This standard specifies the SCSI Architecture Model. The purpose of the architecture is to provide a common basis for the coordination of SCSI standards and to specify those aspects of SCSI I/O system behavior that are independent of a particular technology and common to all implementations.

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which patents, if any, may apply to this standard. As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Contents

Tables

Figures

Foreword

This foreword is not part of American National Standard INCITS 515-201x.

The purpose of this standard is to provide a basis for the coordination of SCSI standards development and to define requirements, common to all SCSI technologies and implementations, that are essential for compatibility with host SCSI application software and device-resident firmware across all SCSI transport protocols. These requirements are defined through a reference model that specifies the behavior and abstract structure that is generic to all SCSI I/O system implementations.

With any technical document there may arise questions of interpretation as new products are implemented. INCITS has established procedures to issue technical opinions concerning the standards developed by INCITS. These procedures may result in SCSI Technical Information Bulletins being published by INCITS.

These Bulletins, while reflecting the opinion of the Technical Committee that developed the standard, are intended solely as supplementary information to other users of the standard. This standard, ANSI INCITS 515-201x, as approved through the publication and voting procedures of the American National Standards Institute, is not altered by these bulletins. Any subsequent revision to this standard may or may not reflect the contents of these Technical Information Bulletins.

Current INCITS practice is to make Technical Information Bulletins available through:

INCITS Online Store   http://www.techstreet.com/INCITS.html
managed by Techstreet   Telephone: 1-734-302-7801 or
1327 Jones Drive   1-800-699-9277
Ann Arbor, MI 48105   Facsimile: 1-734-302-7811

or

Global Engineering   http://global.ihs.com/
15 Inverness Way East   Telephone: 1-303-792-2181 or
Englewood, CO 80112-5704   1-800-854-7179
                                         Facsimile: 1-303-792-2192

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, National Committee for Information Technology Standards, Information Technology Industry Council, Suite 610, 1101 K Street, NW, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by the InterNational Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time of it approved this standard, INCITS had the following members:

Editor's Note 1: <<Insert INCITS member list>>

The INCITS Technical Committee T10 on SCSI Storage Interfaces, that reviewed this standard, had the following members:

Ralph O. Weber, Chair
William Martin, Vice-Chair
John Geldman, Secretary

Organization Represented   *Name of Representative*

Amphenol CorporationGregory McSorley

David Chan (Alt)
Paul Coddington (Alt)
Zhineng Fan (Alt)
Adrian Green (Alt)
Martin Li (Alt)
Chris Lyon (Alt)
Alex Persaud (Alt)
Chansy Phommachanh (Alt)
Michael Wingard (Alt)
CN Wong (Alt)
Matt Wright (Alt)

Avago Technologies ....George Penokie
Patrick Bashford (Alt)
Brad Besmer (Alt)
Srikiran Dravida (Alt)
Bernhard Laschinsky (Alt)
Harvey Newman (Alt)
Robert Sheffield (Alt)
Bill Voorhees (Alt)

Brocade..David Peterson
Scott Kipp (Alt)
Steven Wilson (Alt)

Dell, Inc. ....Kevin Marks
Mark Bokhan (Alt)
Gary Kotzur (Alt)
Bill Lynn (Alt)
Ash McCarty (Alt)
Daniel Oelke (Alt)

EMC Corporation ....Gary Robinson
David Black (Alt)
Erin Bournival (Alt)
George Ericson (Alt)
Mickey Felton (Alt)
Christopher Goonan (Alt)
Marlon Ramroopsingh (Alt)

FCI Electronics..Donald Harper
Brad Brubaker (Alt)
Michael Craton (Alt)
Michael Scholeno (Alt)
Dave Sideck (Alt)

Foxconn Electronics...Fred Fons
Gary Hsieh (Alt)
Glenn Moore (Alt)
Mike Shu (Alt)
Miller Zhao (Alt)

Fujitsu America Inc...Kun Katsumata
Osamu Kimura (Alt)
Mark Malcolm (Alt)
Gene Owens (Alt)
Sandy Wilson (Alt)

Futurewei Technologies Inc...Alan Yoder
  Xiaoyu Ge (Alt)
  Xiaoyan He (Alt)
  Jia Shi (Alt)
  HengLiang Zhang (Alt)
Hewlett-Packard Company ....Curtis Ballard
  Wayne Bellamy (Alt)
  Chris Cheng (Alt)
  Rob Elliott (Alt)
  Joe Foster (Alt)
  Barry Olawsky (Alt)
  Han Wang (Alt)
  Jeff Wolford (Alt)
HGST ....Joe Breher
  David Brewer (Alt)
  Frank Chu (Alt)
  Jason Gao (Alt)
  Chet Mercado (Alt)
  Nadesan Narenthiran (Alt)
  Paul Suhler (Alt)
IBM Corporation ....Kevin Butt
  Mike Osborne (Alt)
Intel Corporation..Pak-Lung Seto
  Richard Mellitz (Alt)
KnowledgeTek Inc..Dennis Moore
  Hugh Curley (Alt)
Marvell Semiconductor Inc...Paul Wassenberg
  Wei Liu (Alt)
  Wei Zhou (Alt)
Micron Technology Inc ....John Geldman
  Jerry Barkley (Alt)
  Andrew Dunn (Alt)
  Neal Galbo (Alt)
  Michael George (Alt)
  Alan Haffner (Alt)
  Daniel Hubbard (Alt)
  Carl Mies (Alt)
  Niels Reimers (Alt)
  Bob Warren (Alt)
Microsoft Corporation...Lee Prewitt
  Douglas King (Alt)
  Paul Luber (Alt)
  Bryan Matthew (Alt)
  Steve Olsson (Alt)
Molex Inc...Jay Neer
  Cong Gao (Alt)
  Alex Haser (Alt)
  Ed Poh (Alt)
  Michael Rost (Alt)
  Darian Schulz (Alt)
NetApp Inc ....Frederick Knight
  Chris Fore (Alt)
  Jaimon George (Alt)
  Ali Yavari (Alt)
OCZ Storage Solutions Inc. ....Tom Friend
  Don Harwood (Alt)

Oracle...Dennis Appleyard
                                                              Jon Allen (Alt)
                                                              Seth Goldberg (Alt)
                                                              Martin Petersen (Alt)
                                                              Phi Tran (Alt)
                                                              Lee Wan-Hui (Alt)
PMC-Sierra ....Tim Symons
                                                              David Allen (Alt)
                                                              Paul Borsetti (Alt)
                                                              Graeme Boyd (Alt)
                                                              Steve Gerson (Alt)
                                                              Bill Lye (Alt)
                                                              Ie-Wei Njoo (Alt)
                                                              Keith Shaw (Alt)
                                                              Gregory Tabor (Alt)
                                                              Neil Wanamaker (Alt)
                                                              Rod Zavari (Alt)
Quantum Corporation...Paul Stone
                                                              Rod Wideman (Alt)
Samsung Semiconductor Inc (SSI)....................................William Martin
                                                              Judy Brock (Alt)
                                                              HeeChang Cho (Alt)
                                                              KeunSoo Jo (Alt)
                                                              Sreenivas Krishnan (Alt)
                                                              Sung Lee (Alt)
                                                              Bhavith M.P. (Alt)
                                                              Truong Nguyen (Alt)
                                                              Aishwarya Ravichandran (Alt)
SanDisk IL Ltd...Avraham Shimor
                                                              Dave Landsman (Alt)
                                                              Yoni Shternhell (Alt)
Seagate Technology..Gerald Houlder
                                                              Michael Connolly (Alt)
                                                              Alvin Cox (Alt)
                                                              Neil Edmunds (Alt)
                                                              Timothy Feldman (Alt)
                                                              John Fleming (Alt)
                                                              Jim Hatfield (Alt)
                                                              Parag Maharana (Alt)
                                                              Alan Westbury (Alt)
                                                              Judy Westby (Alt)
TE Connectivity ....Dan Gorenc
                                                              Tom Grzysiewicz (Alt)
                                                              John Hackman (Alt)
                                                              Kyle Klinger (Alt)
                                                              Joel Meyers (Alt)
                                                              Andy Nowak (Alt)
                                                              Eric Powell (Alt)
                                                              Yasuo Sasaki (Alt)
                                                              Scott Shuey (Alt)
                                                              Robert Wertz (Alt)
Toshiba America Electronic Components Inc......................Mike Fitzpatrick
                                                              Dan Colegrove (Alt)
                                                              Patrick Hery (Alt)
                                                              Yuji Katori (Alt)
                                                              Tom McGoldrick (Alt)
                                                              Scott Wright (Alt)

VMware Inc .... Patrick Dirks
Deepak Babarjung (Alt)
Neil H. MacLean (Alt)
Mike Panas (Alt)
Murali Rajagopal (Alt)
Ahmad Tawil (Alt)
Western Digital Corporation .... Curtis Stevens
Marvin DeForest (Alt)
Michael Koffman (Alt)
Larry McMillan (Alt)
Ralph Weber (Alt)

Introduction

This standard is divided into the following clauses and annexes:

Clause 1 is the scope.
Clause 2 enumerates the normative references that apply to this standard.
Clause 3 describes the definitions, symbols, and abbreviations used in this standard.
Clause 4 describes the overall SCSI architectural model.
Clause 5 describes the SCSI command model element of the SCSI architecture.
Clause 6 describes the events that may be detected by a SCSI device.
Clause 7 describes the task management functions common to SCSI devices.
Clause 8 describes the task set management capabilities common to SCSI devices.
Annex A summarizes the identifier and name definitions of the SCSI transport protocols.
Annex B summarizes the SCSI Initiator Port attributes and SCSI Target Port attributes supported by SCSI transport protocols.
Annex C lists the terminology differences between this standard and previous versions of this standard.
Annex D describes SCSI transport protocol acronyms.
Bibliography lists a bibliography for this standard.

SCSI standards family

Figure 0 shows the relationship of this standard to the other standards and related projects in the SCSI family of standards as of the publication of this standard.
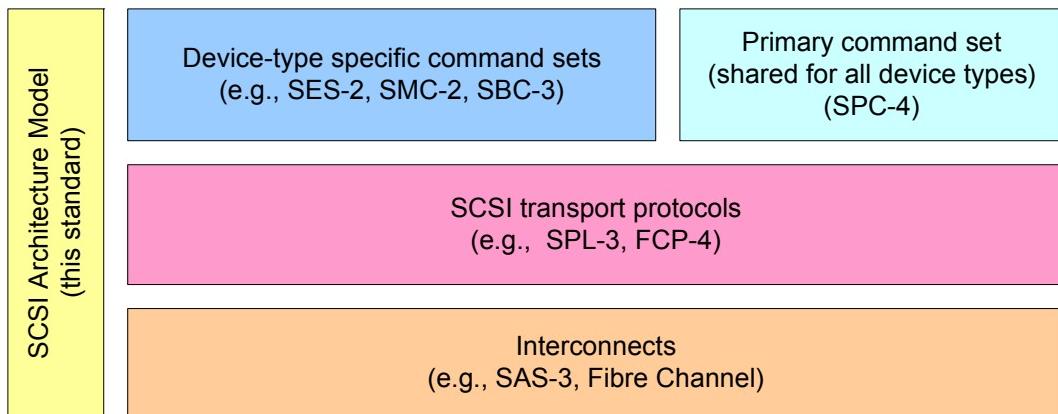
Figure 0 — SCSI document structure

The SCSI document structure in figure 0 is intended to show the general applicability of the documents to one another. Figure 0 is not intended to imply any hierarchy, protocol stack, or system architecture relationship.

The functional areas identified in figure 0 characterize the scope of standards within a group as follows:

SCSI Architecture Model: Defines the SCSI systems model, the functional partitioning of the SCSI standard set and requirements applicable to all SCSI implementations and implementation standards.

Device-Type Specific Command Sets: Implementation standards that define specific device types including a device model for each device type. These standards specify the required commands and behaviors that are specific to a given device type and prescribe the requirements to be followed by a SCSI initiator device when sending commands to a SCSI target device having the specific device type. The commands and behaviors for a specific device type may include by reference commands and behaviors that are defined by other command sets.

Shared Command Set: An implementation standard that defines a model for all SCSI device types. This standard specifies the required commands and behavior that is common to all SCSI devices, regardless of device type, and prescribes the requirements to be followed by a SCSI initiator device when sending commands to any SCSI target device.

SCSI Transport Protocols: Implementation standards that define the requirements for exchanging information so that different SCSI devices are capable of communicating.

Interconnects: Implementation standards that define the communications mechanism employed by the SCSI transport protocols. These standards may describe the electrical and signaling requirements essential for SCSI devices to interoperate over a given interconnect. Interconnect standards may allow the interconnection of devices other than SCSI devices in ways that are outside the scope of this standard.

The term SCSI is used to refer to the family of standards described in this subclause.

**American National Standard
for Information Technology -**

SCSI Architecture Model - 5 (SAM-5)

1 Scope

The set of Small Computer System Interface (SCSI) standards consists of this standard and the SCSI implementation standards described in 4.2. This standard defines a reference model that specifies common behaviors for SCSI devices, and an abstract structure that is generic to all SCSI I/O system implementations.

The set of SCSI standards specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming SCSI implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability.

The following concepts from previous versions of this standard are made obsolete by this standard:

a)   support for the SPI-5 SCSI transport protocol;
b)   Contingent Allegiance;
c)   the TARGET RESET task management function;
d)   basic task management model;
e)   untagged tasks; and
f)   linked command function.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14776-357, *Automation/Drive Interface - Commands - 2 (ADC-2)*

ISO/IEC 14776-454, *SCSI Primary Commands - 4 (SPC-4)* (under consideration)

ISO/IEC 14776-323, *SCSI Block Commands - 3 (SBC-3)* (under consideration)

ISO/IEC 14165-225, *Fibre Channel - Single-Byte Command Code Sets - 5 Mapping Protocol (FC-SB-5)* (under consideration)

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1 additional sense code
combination of the ADDITIONAL SENSE CODE field and the ADDITIONAL SENSE CODE QUALIFIER field in the sense data (see 3.1.104 and SPC-4)

3.1.2 aggregation
when referring to classes (see 3.1.11), a form of association that defines a whole-part relationship between the whole (i.e., aggregate) and its parts

3.1.3 application client
class whose objects are, or an object that is, the source of commands and task management function requests

Note 1 to entry: See 4.6.28.

3.1.4 argument
information provided as input to or output from a procedure call (see 3.1.76)

3.1.5 association
when referring to classes (see 3.1.11), a relationship between two or more classes that specifies connections among their objects

3.1.6 attribute
when referring to classes (see 3.1.11), a named property of a class that describes a range of values that the class or its objects may hold

Note 1 to entry: When referring to objects, a named property of the object.

3.1.7 auto contingent allegiance (ACA)
task set condition established following the completion of a command with CHECK CONDITION status when the NACA bit is set to one in the CONTROL byte

Note 1 to entry: See 5.9.

3.1.8 background operation
operation started by a command that continues processing after the command is no longer in the task set

Note 1 to entry: See 5.5.

3.1.9 blocking boundary
task set boundary denoting a set of conditions that inhibit commands outside the boundary from requesting to be processed by the device server

Note 1 to entry: See 8.8.

3.1.10 byte
8-bit construct

3.1.11 class
description of a set of objects that share the same attributes, operations, relationships and semantics

Note 1 to entry: Classes may have attributes and may support operations.

Note 2 to entry: Examples of class relationships are aggregation, association, generalization, and dependency.

3.1.12 class diagram
shows a set of classes and their relationships

Note 1 to entry: Class diagrams are used to illustrate the static design view of a system.

Note 2 to entry: See 3.6.3.

3.1.13 client
entity that requests a service from a server

Note 1 to entry: This standard defines one client, the application client.

3.1.14 client-server
relationship established between a pair of entities where one (the client) requests the other (the server) to perform some operation or unit of work on the client's behalf

Note 1 to entry: See 4.4.

3.1.15 command
request describing a unit of work to be performed by a device server

3.1.16 command completion
device server or task manager has sent a **Send Command Complete** SCSI transport protocol service response for the command (see 5.4.2.4)

3.1.17 command descriptor block (CDB)
structure used to communicate a command from an application client to a device server having a fixed length of 6 bytes, 10 bytes, 12 bytes, or 16 bytes, or a variable length of between 12 and 260 bytes

Note 1 to entry: See 5.2 and SPC-4.

3.1.18 command duration limit
maximum duration of command processing specified by an application client

Note 1 to entry: See 8.6.

3.1.19 command identifier
numerical identifier of a command (see 3.1.15)

Note 1 to entry: See 4.6.31.3.

3.1.20 command priority
relative scheduling importance of a command having the SIMPLE task attribute among the set of commands having the SIMPLE task attribute already in the task set

Note 1 to entry: See 8.5.

3.1.21 command standard
SCSI standard that defines the model, commands, and parameter data for a device type (e.g., SPC-4, SBC-3)

Note 1 to entry: See clause 1.

3.1.22 completed command
command that has completed with a service response of COMMAND COMPLETE

3.1.23 confirmation
acknowledge returned to an application client or device server that signals the completion of a service request

3.1.24 constraint
when referring to classes (see 3.1.11) and objects, a mechanism for specifying semantics or conditions that are maintained as true between entities

Note 1 to entry: An example of a constraint is a required condition between associations.

3.1.25 copy manager
class whose objects are each, or an object that is, an application client that processes third-party copy commands and manages copy operations (see SPC-4) from within a logical unit

Note 1 to entry: See 4.6.20.

3.1.26 current command
command that has a data transfer SCSI transport protocol service request in progress (see 5.4.3) or is in the process of sending command status

Note 1 to entry: Each SCSI transport protocol standard may define the SCSI transport protocol specific conditions under which a command is considered a current command.

3.1.27 deferred error
error generated by a background operation (see SPC-4)

3.1.28 dependency
relationship between two elements in which a change to one element (e.g., the server) may affect or supply information needed by the other element (e.g., the client)

3.1.29 dependent logical unit
logical unit that is addressed via some other logical unit(s) in a hierarchical LUN structure

Note 1 to entry: See 4.6.18.4.

3.1.30 device model
description of a type of SCSI target device

Note 1 to entry: Examples of SCSI target devices are a block device and a stream device.

3.1.31 device server
class whose objects process, or an object that processes, commands according to the requirements for command management described in clause 8

Note 1 to entry: See 4.6.19.

3.1.32 device service request
request submitted by an application client conveying a command to a device server

3.1.33 device service response
response returned to an application client by a device server on completion of a command

3.1.34 extended logical unit addressing
logical unit addressing method

Note 1 to entry: See 4.7.7.5.

3.1.35 faulted I_T nexus
I_T nexus on which a command was terminated with CHECK CONDITION status that resulted in the establishment of an ACA

Note 1 to entry: The faulted I_T nexus condition is cleared when the ACA condition is cleared

3.1.36 3.1.37 field
group of one or more contiguous bits, part of a larger structure (e.g., a CDB (see 3.1.17) or sense data (see 3.1.104))

3.1.38 generalization
when referring to classes (see 3.1.11), a relationship among classes where one class (i.e., superclass) shares the attributes and operations of one or more classes (i.e., subclasses)

3.1.39 hard reset
condition resulting from a power on condition or a reset event in which the SCSI device performs the hard reset operations described in 6.3.2, SPC-4, and the appropriate command standards

3.1.40 hierarchical LUN structure
inverted tree structure for forming and parsing LUNs (see 3.1.59) containing up to four addressable levels (see 4.6.11)

Note 1 to entry: See 4.7.6.3.

3.1.41 hierarchical logical unit
logical unit that is addressed by a LUN in a hierarchical LUN structure

Note 1 to entry: See 4.6.15.

3.1.42 I_T nexus
nexus between a SCSI initiator port and a SCSI target port

Note 1 to entry: See 4.6.32.2.

3.1.43 I_T nexus loss
condition resulting from a hard reset condition or an I_T nexus loss event in which the SCSI device performs the I_T nexus loss operations described in 6.3.4, SPC-4, and the appropriate command standards

3.1.44 I_T nexus loss event
SCSI transport protocol specific event that results in an I_T nexus loss condition as described in 6.3.4

3.1.45 I_T nexus transaction
information transferred between SCSI ports in a single data structure with defined boundaries (e.g., an information unit)

3.1.46 I_T_L nexus
nexus between a SCSI initiator port, a SCSI target port, and a logical unit

Note 1 to entry: See 4.6.32.3.

3.1.47 identifier
label of an object that is unique within a specified context and that may change

3.1.48 implicit head of queue
optional processing model for specific commands wherein a command may be treated as if it had been received with a HEAD OF QUEUE task attribute

Note 1 to entry: See 8.2.

3.1.49 incorrect logical unit number
logical unit number of a logical unit that does not exist in the SCSI target device when addressed through a given I_T nexus

Note 1 to entry: See 4.7.1.

3.1.50 initiator port identifier
value by which a SCSI initiator port is referenced within a SCSI domain

Note 1 to entry: See 4.6.8.2.

3.1.51 initiator port name
name (see 3.1.64) of a SCSI initiator port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI initiator port

Note 1 to entry: The name may be made available to other SCSI devices or SCSI ports in that SCSI domain in SCSI transport protocol specific ways.

Note 2 to entry: See 4.6.8.3.

3.1.52 instance
concrete manifestation of an abstraction to which a set of operations may be applied and which may have a state that stores the effects of the operation

Note 1 to entry: An object is an instance of a class.

3.1.53 in transit
information that has been delivered to a service delivery subsystem for transmission, but not yet arrived at the intended recipient

3.1.54 layer
subdivision of the architecture constituted by SCSI initiator device and SCSI target device elements at the same level relative to the interconnect

3.1.55 link
individual connection between two objects in an object diagram representing an instance of a relationship

3.1.56 logical unit
class whose objects implement, or an object that implements, a device model that manages and processes commands sent by an application client

Note 1 to entry: See 4.6.18.

3.1.57 logical unit inventory
list of LUNs reported by a REPORT LUNS command (see SPC-4)

3.1.58 logical unit name
name (see 3.1.64) of a logical unit that is world wide unique within the SCSI transport protocol of a SCSI domain in which the SCSI device containing the logical unit has SCSI ports (see 4.6.4.2)

Note 1 to entry: The logical unit name may be made available to other SCSI devices or SCSI ports in SCSI transport protocol specific ways.

3.1.59 logical unit number (LUN)
for transport protocol standards, 64-bit or 16-bit identifier for a logical unit and for command standards, 64-bit identifier for a logical unit.

Note 1 to entry: See 4.7.

3.1.60 logical unit reset
condition resulting from a hard reset condition or a logical unit reset event in which the logical unit performs the logical unit reset operations described in 6.3.3, SPC-4, and the appropriate command standards

3.1.61 logical unit reset event
event that results in a logical unit reset condition as described in 6.3.3

3.1.62 management logical unit
class whose objects are each, or an object that is, a logical unit that only performs management functions

Note 1 to entry: Management logical units allow an application client to issue requests to receive specific information and manage specific information relating to a SCSI target device.

Note 2 to entry: See 4.6.16.

3.1.63 multiplicity
when referring to classes (see 3.1.11), an indication of the range in number of allowable instances that a class or an attribute may have

3.1.64 name
label of an object that is unique within a defined context and should never change

3.1.65 nexus
I_T nexus or I_T_L nexus

3.1.66 non-faulted I_T nexus
I_T nexus that is not a faulted I_T nexus (see 3.1.35)

3.1.67 object
entity with a well-defined boundary and identity that encapsulates state and behavior

Note 1 to entry: All objects are instances of classes (see 3.1.52).

3.1.68 object diagram
shows a set of objects and their relationships at a point in time

Note 1 to entry: Object diagrams are used to illustrate static shapshots of instances of the things found in class diagrams.

Note 2 to entry: See 3.6.4.

3.1.69 operation
when referring to classes and objects, a service that may be requested from any object of the class in order to affect behavior

Note 1 to entry: Operations describe what a class is allowed to do (see 3.6).

3.1.70 peer entities
entities within the same layer (see 3.1.54)

3.1.71 power loss expected
condition resulting from a power loss expected event in which the logical unit performs the power loss expected operations described in 6.3.5, SPC-4, and the appropriate transport protocol and command standards

3.1.72 power loss expected event
event that results in a power loss expected condition (see 3.1.71) as described in 6.3.5

3.1.73 power on
condition resulting from a power on event in which the SCSI device performs the power on operations described in 6.3.1, SPC-4, and the appropriate command standards

3.1.74 power on event
power being applied to a SCSI device, resulting in a power on condition as described in 6.3.1

3.1.75 procedure
operation that is invoked through an external calling interface

3.1.76 procedure call
model used by this standard for the interfaces involving both the SAL (see 3.1.85) and STPL (see 3.1.97), having the appearance of a programming language function call

Note 1 to entry: See 3.9.

3.1.77 protocol
specification and/or implementation of the requirements governing the content and exchange of information passed between distributed entities through a service delivery subsystem

3.1.78 queue
class whose objects maintain, or an object that maintains, an arrangement of commands within a task set (see 3.1.123)

3.1.79 receiver
client or server that is the recipient of a service delivery transaction

3.1.80 reference model
standard model used to specify system requirements in an implementation independent manner

3.1.81 relative port identifier
identifier for a SCSI port that is unique within a SCSI device

Note 1 to entry: See 4.6.5.2.

3.1.82 request-response transaction
interaction between a pair of distributed, cooperating entities, consisting of a request for service submitted to
an entity followed by a response conveying the result

3.1.83 reset event
SCSI transport protocol specific event that results in a hard reset condition as described in 6.3.2

3.1.84 role
when referring to classes (see 3.1.11) and objects, a label at the end of an association or aggregation that
defines a relationship to the class on the other side of the association or aggregation.

3.1.85 SCSI application layer (SAL)
protocols and procedures that implement or issue commands and task management functions by using
services provided by a STPL (see 3.1.97)

3.1.86 SCSI device
class whose objects are, or an object that is, connected to a service delivery subsystem and supports a SCSI
application protocol

Note 1 to entry: See 4.6.4.

3.1.87 SCSI device name
name (see 3.1.64) of a SCSI device that is world wide unique within the SCSI transport protocol of a SCSI
domain in which the SCSI device has SCSI ports (see 4.6.4.2)

Note 1 to entry: The SCSI device name may be made available to other SCSI devices or SCSI ports in SCSI
transport protocol specific ways.

3.1.88 SCSI domain
I/O system consisting of a set of SCSI devices and a service delivery subsystem, where the SCSI devices
interact with one another by means of the service delivery subsystem

3.1.89 SCSI event

condition defined by this standard that is detected by a SCSI device and that requires notification of its
occurrence within the SCSI device

Note 1 to entry: See clause 6.

Note 2 to entry: An example of a SCSI event is a logical unit reset.

3.1.90 SCSI I/O operation
operation defined by a command or a task management function

3.1.91 SCSI I/O system
I/O system, consisting of two or more SCSI devices, a SCSI interconnect, and a SCSI transport protocol that
collectively interact to perform SCSI I/O operations

3.1.92 SCSI initiator device
class whose objects originate, or an object that originates, device service and task management requests to
be processed by a SCSI target device and receives device service and task management responses from
SCSI target devices

3.1.93 SCSI initiator port

class whose objects each act, or an object that acts, as the connection between application clients and a service delivery subsystem through which server requests and server responses are routed

Note 1 to entry: See 4.6.8.

3.1.94 SCSI port

class whose objects connect, or an object that connects, the application client, device server or task manager to a service delivery subsystem through which server requests and server responses are routed

Note 1 to entry: A SCSI port is either a SCSI initiator port (see 3.1.93), a SCSI target port (see 3.1.96), or both.

Note 2 to entry: See 4.6.5.

3.1.95 SCSI target device

class whose objects receive, or an object that receives, device service requests and task management requests for processing and sends device service responses and task management responses to SCSI initiator devices

Note 1 to entry: See 4.6.9.

3.1.96 SCSI target port

class whose objects each act, or an object that acts, as the connection between device servers and task managers and a service delivery subsystem through which server requests and server responses are routed

Note 1 to entry: See 4.6.6.

3.1.97 SCSI transport protocol layer (STPL)

protocol and services used by a SAL (see 3.1.85) to transport data representing a SCSI application protocol transaction

3.1.98 SCSI transport protocol service confirmation

procedure call from the STPL notifying the SAL that a SCSI transport protocol service request has completed

3.1.99 SCSI transport protocol service indication

procedure call from the STPL notifying the SAL that a SCSI transport protocol transaction has occurred

3.1.100 SCSI transport protocol service request

procedure call to the STPL to begin a SCSI transport protocol service transaction

3.1.101 SCSI transport protocol service response

procedure call to the STPL containing a reply from the SAL in response to a SCSI transport protocol service indication

3.1.102 SCSI transport protocol specific

implementation of the referenced item is defined by a SCSI transport protocol standard (see clause 2 and Bibliography)

3.1.103 sender

client or server that originates a service delivery transaction

3.1.104 sense data

data describing command completion information that a device server delivers to an application client in the same I_T nexus transaction as the status or as parameter data in response to a REQUEST SENSE command

Note 1 to entry: See 5.13 and SPC-4.

3.1.105 sense key
SENSE KEY field in the sense data (see 3.1.104 and SPC-4)

3.1.106 server
entity that performs a service on behalf of a client

3.1.107 server request
transaction from a client to a server invoking a service

3.1.108 server response
transaction from a server to a client conveying the result of a request

3.1.109 service
any operation or function performed by a SCSI object that is invoked by other SCSI objects

3.1.110 service delivery failure
any non-recoverable error causing the corruption or loss of one or more service delivery transactions while in transit

3.1.111 service delivery subsystem
class whose objects are, or an object that is, part of a SCSI I/O system that transmits service requests to a logical unit or SCSI target device and returns logical unit or SCSI target device responses to a SCSI initiator device

Note 1 to entry: See 4.6.3.

3.1.112 service delivery transaction
request or response sent through a service delivery subsystem

3.1.113 standard INQUIRY data
data returned to an application client as a result of an INQUIRY command (see SPC-4) with the EVPD bit set to zero

Note 1 to entry: Fields in the standard INQUIRY data are referenced by name in this standard.

3.1.114 target port identifier
value by which a SCSI target port is referenced within a SCSI domain

Note 1 to entry: See 4.6.6.2.

3.1.115 target port name
name (see 3.1.64) of a SCSI target port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI target port

Note 1 to entry: The name may be made available to other SCSI devices or SCSI ports in that SCSI domain in SCSI transport protocol specific ways.

Note 2 to entry: See 4.6.6.3.

3.1.116 task
synonymous with command (see 3.1.15 and Annex C)

3.1.117 task attribute
attribute of a command (see 3.1.15) that specifies the processing relationship of a command with regard to other commands in the task set (see 3.1.123)

Note 1 to entry: See 8.4.

3.1.118 task management function
task manager service capable of being requested by an application client to affect the processing of one or more commands

Note 1 to entry: See clause 7.

3.1.119 task management request
request submitted by an application client, invoking a task management function to be processed by a task manager

3.1.120 task management response
response returned to an application client by a task manager in reply to a task management request

3.1.121 task manager
class whose objects control, or an object that controls, the sequencing of commands and processes task management functions

Note 1 to entry: See 4.6.21.

3.1.122 task router
class whose objects route, or an object that routes, commands and task management functions between a service delivery subsystem (see 3.1.111) and the appropriate task manager(s)

Note 1 to entry: See 4.6.7.

3.1.123 task set
class whose objects are, or an object that is, a group of commands within a logical unit, whose interaction is dependent on the task management (e.g., queuing) and ACA requirements

Note 1 to entry: See 4.6.22.

3.1.124 task tag
term used by previous versions of this standard (see Annex C)

Note 1 to entry: Synonymous with command identifier (see 3.1.19).

3.1.125 token
representation of a collection of data

3.1.126 transaction
cooperative interaction between two entities, involving the exchange of information or the processing of some request by one entity on behalf of the other

3.1.127 transport protocol standard
SCSI standard that defines a transport protocol (e.g., SPL-3)

3.1.128 3.1.129 well known logical unit
class whose objects are each, or an object that is, a logical unit that only performs specific functions

Note 1 to entry: Well known logical units allow an application client to issue requests to receive and manage specific information relating to a SCSI target device.

Note 2 to entry: See 4.6.17.

3.1.130 well known logical unit number (W-LUN)
LUN that identifies a well known logical unit

Note 1 to entry: See 4.7.7.5.1.

3.2 Symbols and Acronyms

3.2.1 Abbreviations

Abbreviations used in this standard:

Abbreviation	Meaning
ACA	Auto Contingent Allegiance (see 3.1.7)
ADC-2	Automation/Drive Interface - Commands - 2 (see clause 2)
ADT-2	Automation/Drive Interface Transport Protocol - 2 (see Bibliography)
CDB	Command Descriptor Block (see 3.1.17)
CRN	Command Reference Number
FCP-4	Fibre Channel Protocol - 4 (see Bibliography)
FC-SB-5	Fibre Channel - Single-Byte Command Code Sets - 5 Mapping Protocol (see clause 2)
iSCSI	Internet SCSI (see Bibliography)
ISO	Organization for International Standardization
LUN	Logical Unit Number (see 3.1.59)
n/a	Not Applicable
RAID	Redundant Array of Independent Disks
ROD	Representation Of Data (see SPC-4)
SAL	SCSI application layer (see 3.1.85)
SAM-2	SCSI Architecture Model - 2
SAM-3	SCSI Architecture Model - 3
SAM-4	SCSI Architecture Model - 4
SAS-3	Serial Attached SCSI - 3 (see Bibliography)
SBC-3	SCSI Block Commands - 3 (see clause 2)
SBP-3	Serial Bus Protocol - 3 (see Bibliography)
SCSI	The architecture defined by the family of standards described in clause 2
SPC-4	SCSI Primary Commands - 4 (see clause 2)
SRP	SCSI RDMA Protocol (see Bibliography)
STPL	SCSI transport protocol layer (see 3.1.97)
UML	Unified Modeling Language
VPD	Vital Product Data (see SPC - 4)
W-LUN	Well known logical unit number (see 3.1.130)

3.2.2 Units

Units used in this standard:

Unit	Meaning
ms	millisecond (i.e., 10^{-3} seconds)

3.3 Keywords

3.3.1 invalid
keyword used to describe an illegal or unsupported bit, byte, word, field, or code value

Note 1 to entry: Receipt by a device server of an invalid bit, byte, word, field, or code value shall be reported as an error.

3.3.2 mandatory
keyword indicating an item that is required to be implemented as defined in this standard

3.3.3 may
keyword that indicates flexibility of choice with no implied preference

Note 1 to entry: May is synonymous with the phrase "may or may not".

3.3.4 may not
keyword that indicates flexibility of choice with no implied preference

Note 1 to entry: May not is synonymous with the phrase "may or may not".

3.3.5 obsolete
keyword indicating that an item was defined in prior SCSI standards but has been removed from this standard

3.3.6 option, optional
keywords that describe features that are not required to be implemented by this standard

Note 1 to entry: If any optional feature defined by this standard is implemented, then it shall be implemented as defined in this standard.

3.3.7 prohibited
keyword used to describe a feature, function, or coded value that is defined in a non-SCSI standard (i.e., a standard that is not a member of the SCSI family of standards) to which this standard makes a normative reference where the use of said feature, function, or coded value is not allowed for implementations of this standard

3.3.8 reserved
keyword referring to bits, bytes, words, fields, and code values that are set aside for future standardization

Note 1 to entry: A reserved bit, byte, word, or field shall be set to zero, or in accordance with a future extension to this standard.

Note 2 to entry: Recipients are not required to check reserved bits, bytes, words, or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error.

3.3.9 restricted
keyword referring to bits, bytes, words, and fields that are set aside for other identified standardization purposes

Note 1 to entry: A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word, or field in the context where the restricted designation appears.

3.3.10 shall
keyword indicating a mandatory requirement

Note 1 to entry: Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.3.11 should
keyword indicating flexibility of choice with a strongly preferred alternative

Note 1 to entry: Equivalent to the phrase "it is strongly recommended".

3.3.12 vendor specific
specification of the referenced item is determined by the SCSI device vendor

Note 1 to entry: Specification of the referenced item is determined by the SCSI device vendor and may be used differently in various implementations.

3.4 Editorial conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in the glossary or in the text where they first appear.

Upper case is used when referring to the name of a numeric value defined in this specification or a formal attribute possessed by an entity. When necessary for clarity, names of objects, procedure calls, arguments or discrete states are capitalized or set in bold type. Names of fields are identified using small capital letters (e.g., NACA bit).

Names of procedure calls are identified by a name in bold type, such as **Execute Command** (see clause 5). Names of arguments are denoted by capitalizing each word in the name. For instance, Sense Data is the name of an argument in the **Execute Command** procedure call.

Quantities having a defined numeric value are identified by large capital letters. CHECK CONDITION, for example, refers to the numeric quantity defined in table 42 (see 5.3.1). Quantities having a discrete but unspecified value are identified using small capital letters. As an example, COMMAND COMPLETE, indicates a quantity returned by the **Execute Command** procedure call (see clause 5). Such quantities are associated with an event or indication whose observable behavior or value is specific to a given implementation standard.

Lists sequenced by lowercase or uppercase letters show no ordering relationship between the listed items.

EXAMPLE 1 - The following list shows no relationship between the named items:

  a) red (i.e., one of the following colors):
    A) crimson; or
    B) amber;
  b) blue; or
  c) green.

Lists sequenced by numbers show an ordering relationship between the listed items.

EXAMPLE 2 - The following list shows an ordered relationship between the named items:

  1) top;
  2) middle; and
  3) bottom.

If a conflict arises between text, tables, or figures, then the order of precedence to resolve the conflicts is:

  1) text;
  2) tables; and
  3) figures.

Not all tables or figures are fully described in the text. Tables show data format and values.

Notes and examples do not constitute any requirements.

Notes are numbered consecutively throughout this standard.

3.5 Numeric and character conventions

3.5.1 Numeric conventions

A binary number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Underscores or spaces may be included in binary number representations to increase readability or delineate organizational boundaries (e.g., 00010101 11001110b, 00010101_11001110b, 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 to 9 and/or the upper-case English letters A to F immediately followed by a lower-case h (e.g., FA23h). Underscores or spaces may be included in hexadecimal number representations to increase readability or delineate organizational boundaries (e.g., 3456FDCA 84BD5E7Ah, 3456FDCA_84BD5E7Ah, B FD8C FA23h, or B_FD8C_FA23h).

A decimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 to 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

This standard uses the following conventions for representing decimal numbers:

    a)   the decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
    b)   the thousands separator (i.e., separating groups of three digits in a portion of the number) is a space;
    c)   the thousands separator is used in both the integer portion and the fraction portion of a number; and
    d)   the decimal representation for a year is 1999 not 1 999.

Table 1 shows some examples of decimal numbers using various conventions.

Table 1 — Numbering conventions

French	English	This standard
0,6	0.6	0.6
3,141 592 65	3.14159265	3.141 592 65
1 000	1,000	1 000
1 323 462,95	1,323,462.95	1 323 462.95

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g., 666.$\overline{6}$ means 666.666 666... or 666 2/3, and 12.142 $\overline{857}$ means 12.142 857 142 857... or 12 1/7).

A range of numeric values is represented in this standard in the form "a to z", where a is the first value included in the range, all values between a and z are included in the range, and z is the last value included in the range (e.g., the representation "0h to 3h" includes the values 0h, 1h, 2h, and 3h).

3.5.2 Byte encoded character strings conventions

When this standard requires one or more bytes to contain specific encoded characters, the specific characters are enclosed in single quotation marks. The single quotation marks identify the start and end of the characters that are required to be encoded but are not themselves to be encoded. The characters that are to be encoded are shown in the case that is to be encoded.

An ASCII space character (i.e., 20h) may be represented in a string by the character '¬' (e.g., 'SCSI¬device').

The encoded characters and the single quotation marks that enclose them are preceded by text that specifies the character encoding methodology and the number of characters required to be encoded.

EXAMPLE - Using the notation described in this subclause, stating that eleven ASCII characters 'SCSI device' represent encoded characters is the same as writing out the following sequence of byte values: 53h 43h 53h 49h 20h 64h 65h 76h 69h 63h 65h.

3.6 UML notation conventions

3.6.1 Notation conventions overview

This standard uses class diagrams and object diagrams with notation that is based on the Unified Modeling Language (UML).

See 3.6.3 for the conventions used for class diagrams.

See 3.6.4 for the conventions used for object diagrams.

3.6.2 Constraint and note conventions

Class diagrams and object diagrams may include constraints, which specify requirements, and notes, which are informative.

Table 2 shows the notation used for constraints and notes.

Table 2 — Constraint and note notation

Notation	Description
{Constraint text}	The presence of the curly brackets (i.e., {}) defines a constraint that is a normative requirement. An example of a constraint is shown in figure 2.
note text	The absence of curly brackets defines a note that is informative. An example of a note is shown in figure 3.

3.6.3 Class diagram conventions

Table 3 shows the notation used for classes in class diagrams.

Table 3 — Class diagram notation for classes
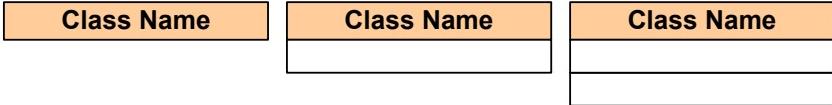
Notation	Description
Class Name **Class Name** **Class Name**	A class with no attributes or operations.
Class Name Attribute01[1] Attribute02[1] **Class Name** Attribute01[1] Attribute02[1]	A class with attributes and no operations.
Class Name Operation01() Operation02()	A class with operations and no attributes.
Class Name Attribute01[1] Attribute02[1] Operation01() Operation02()	A class with attributes and operations.
Class Name Attribute01[1..*] Attribute02[1] Operation01() Operation02()	A class with attributes that have a specified multiplicity (see table 4) and operations.
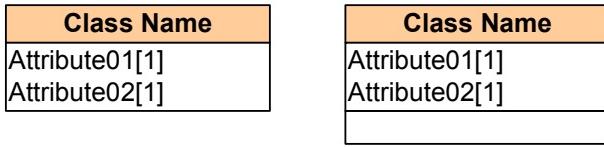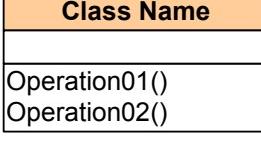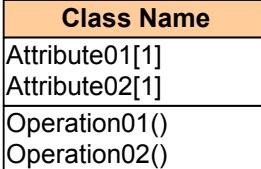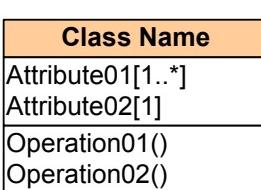
Table 4 shows the notation used to indicate multiplicity of instances of classes and attributes in class diagrams.

Table 4 — Multiplicity notation

Notation	Description
	The number of instances of a class or an attribute is not specified.
1	One instance of the class or attribute exists.
0..*	Zero or more instances of the class or attribute exist.
1..*	One or more instances of the class or attribute exist.
0..1	Zero or one instance of the class or attribute exists.
n..m	n to m instances of the class or attribute exist (e.g., 2..8).
x, n..m	Multiple disjoint instances of the class or attribute exist (e.g., 2, 8..15).
[a] See figure 1 and figure 2 for examples of multiplicity notation.	

Table 5 shows the notation used to denote association (i.e., "knows about") relationships between classes.

Unless the two classes in an association relationship also have an aggregation relationship, association relationships have a multiplicity notation (see table 4) at each end of the relationship line.

Table 5 — Class diagram notation for associations
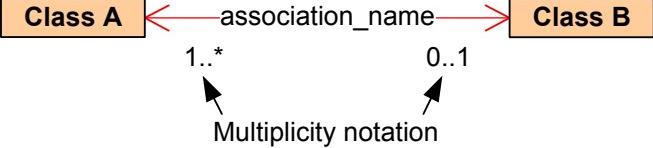
Notation	Description
Class A ← association_name → Class B 1..* 0..1 Multiplicity notation	Class A knows about Class B (i.e., read as "Class A association_name Class B") and Class B knows about Class A (i.e., read as "Class B association_name Class A").
Class A ← Class B 1 0..1	Class B knows about Class A (i.e., read as "Class B knows about Class A") but Class A does not know about Class B.
role name Class A → Class B 0..* 0..1	Class A knows about Class B (i.e., read as "Class A uses the role name attribute of Class B") but Class B does not know about Class A.
Note - The use of role names and association names are optional.	

See figure 1 for examples of association relationships between classes.
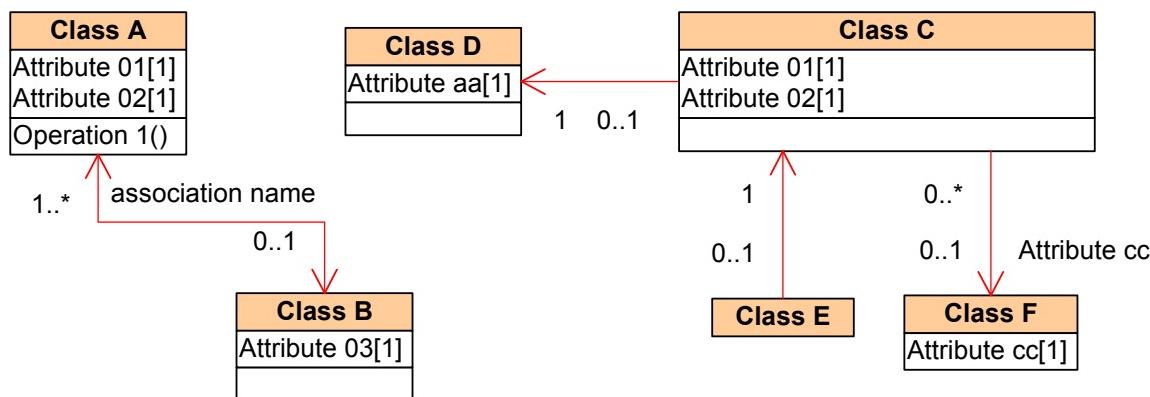
Figure 1 — Examples of association relationships in class diagrams

Table 6 shows the notation used to denote aggregation (i.e., "is a part of" or "contains") relationships between classes. The aggregation relationship is a specific type of association (see table 5) and always include multiplicity notation (see table 4) at each end of the relationship line.

Table 6 — Class diagram notation for aggregations

Notation	Description
	The Part class is part of the Whole class (i.e, read as "the whole contains the part") and may continue to exist even if the Whole class is removed .
	The Part class is part of the Whole class, shall only belong to one Whole class (i.e., read as "the whole contains the part"), and shall not continue to exist if the Whole class is removed (i.e., read as "the whole contains the part").

See figure 2 for examples of aggregation relationships between classes.
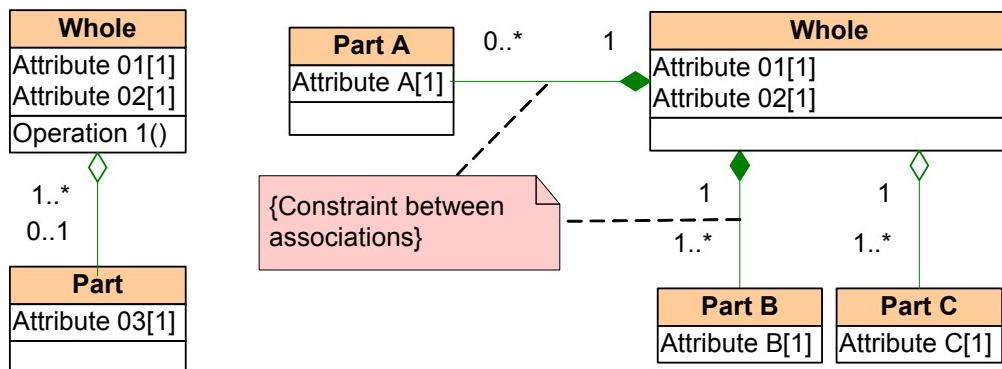
Figure 2 — Examples of aggregation relationships in class diagrams

Table 7 shows the notation used to denote generalization (i.e., "is a kind of") relationships between classes.

Table 7 — Class diagram notation for generalizations

Notation	Description
	Subclass is a kind of superclass. A subclass shares all the attributes and operations of the superclass (i.e., the subclass inherits from the superclass). Inherited attributes are not duplicated in UML drawings.

See figure 3 for examples of generalization relationships between classes.

Single superclass/single subclass:

| **Superclass** |
| Attribute 01[1] |
| Attribute 02[1] |

| **Subclass A** |
| Attribute 03[1] |
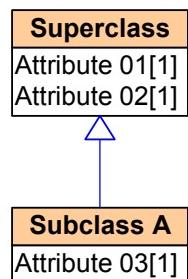
Multiple superclasses/single subclass
(i.e., muliple inheritance):

| **Superclass A** |
| Attribute 1A[1] |
| Attribute 2A[1] |

| **Superclass B** |
| Attribute 1B[1] |
| Attribute 2B[1] |

| **Subclass B** |
| Attribute 04[1] |
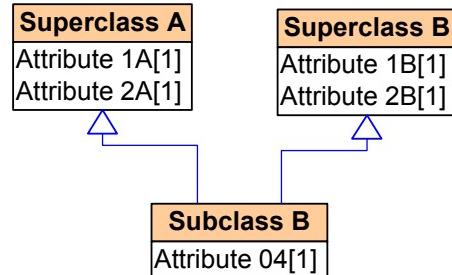
Single superclass/multiple subclasses:

| **Superclass** |
| Attribute 01[1] |
| Attribute 02[1] |

| **Subclass A** | **Subclass B** | **Subclass C** |
| Attribute A[1] | Attribute B[1] | Attribute C[1] |

There is no significance to generalizations that are combined or not combined.
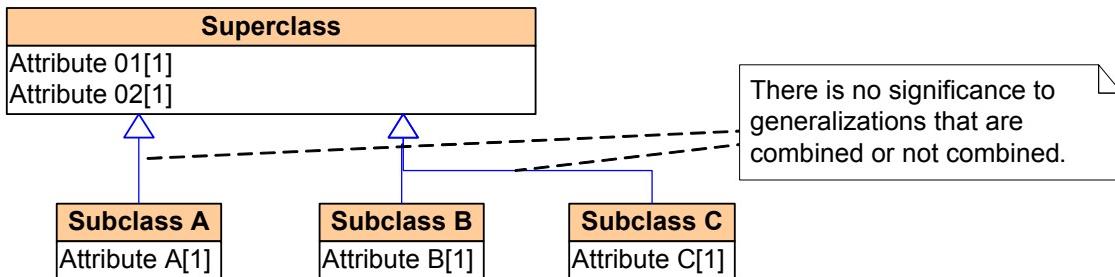
Figure 3 — Example of generalization relationships in class diagrams

Table 8 shows the notation used to denote dependency (i.e., "depends on") relationships between classes.

Table 8 — Class diagram notation for dependency

Notation	Description
Class A ----------> Class B	Class A depends on class B. A change in class B may cause a change in class A.

See figure 4 for an example of a dependency relationship between classes.

Dependent --------------------> Independent

Figure 4 — Example of a dependency relationship in class diagrams

3.6.4 Object diagram conventions

Table 9 shows the notation used for objects in object diagrams.

Table 9 — Object diagram notation for objects
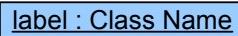
Notation	Description
label : Class Name	Notation for a named object with no attributes.
label : Class Name Attribute01 = x Attribute02 = y	Notation for a named object with attributes.
: Class Name	Notation for an anonymous object with no attributes.
: Class Name Attribute01 = x Attribute02 = y	Notation for an anonymous object with attributes.
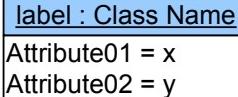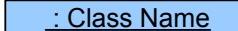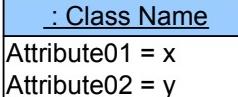
Table 10 shows the notation used to denote link relationships between objects.

Table 10 — Object diagram notation for link

Notation	Description
: Object A ———————— : Object B	An instance of an association between object A and object B.

See figure 5 for examples of a link relationships between objects.

: Class A
Attribute 01 = round
Attribute 02 = red

: Class Aa
Attribute 01 = true
Attribute 02 = 902

O1 : Class a
Attribute 03 = soft

A1 : Class aa
Attribute aa = rain

B1 : Class bb

C1 : Class cc
Attribute cc = USA
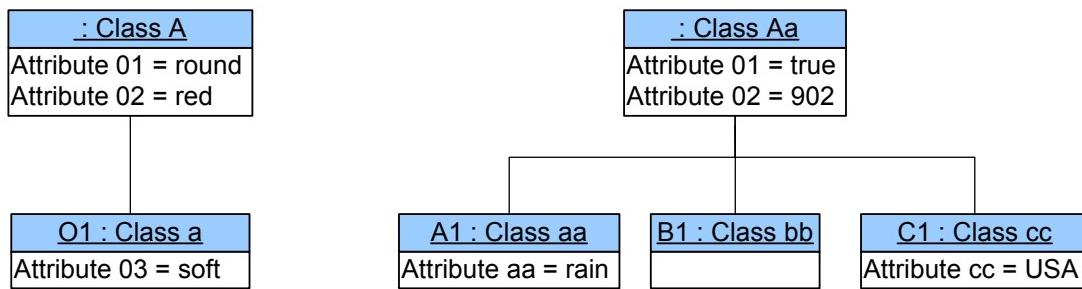
Figure 5 — Examples of link relationships for object diagrams

3.7 State machine conventions

3.7.1 State machine conventions overview

Figure 6 shows how state machines are described.
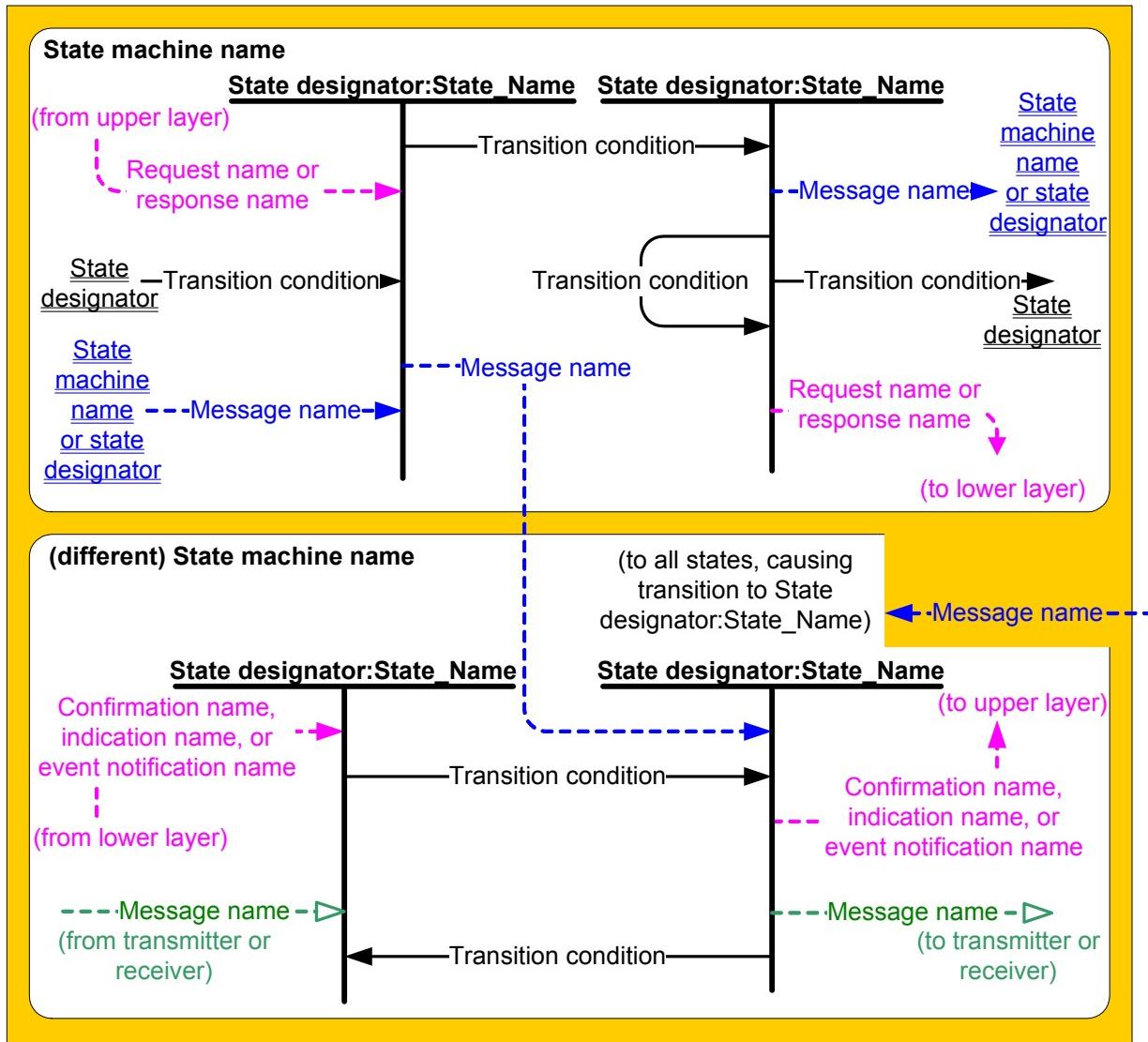
Figure 6 — State machine conventions

Figure 7 shows an example description for one state machine.
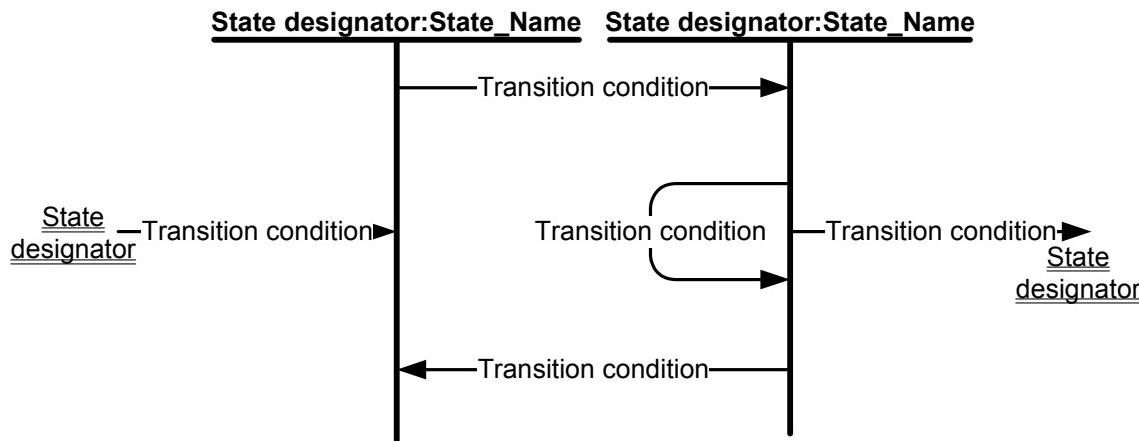
Figure 7 — Example description for one state machine

When multiple state machines are present in a figure, they are enclosed in boxes with rounded corners.

Each state is identified by a state designator and a state name. The state designator (e.g., SL1) is unique among all state machines in this standard. The state name (e.g., Idle) is a brief description of the primary action taken during the state, and the same state name may be used by other state machines. Actions taken while in each state are described in the state description text.

3.7.2 Transitions

Transitions between states are shown with solid lines, with an arrow pointing to the destination state. A transition may be labeled with a transition condition label (i.e., a brief description of the event or condition that causes the transition to occur).

If the state transition exits a figure or enters a figure, then the transition label goes to or from a state designator label with double underlines rather than to or from a state.

The conditions and actions are described fully in the transition description text. In case of a conflict between a figure and the text, the text shall take precedence.

Transitions between states are instantaneous.

Upon entry into a state, all actions to be processed in that state are processed. If a state is re-entered from itself, all actions to be processed in the state are processed again. A state may be entered and exited in zero time if the conditions for exiting the state are valid upon entry into the state.

3.7.3 Messages, requests, indications, confirmations, responses, and event notifications

Messages passed between state machines are shown with dashed lines labeled with a message name. When messages are passed between state machines, they are identified by either:

   a) a dashed line to or from a state machine name label with double underlines and/or state name label with double underlines, if the destination is in a different figure from the source;
   b) a dashed line to or from a state in another state machine in the same figure; or
   c) a dashed line from a state machine name label with double underlines to a "(to all states)" label, if the destination is every state in the state machine.

Requests, indications, confirmations, responses, and event notifications are shown with curved dashed lines originating from or going to the top or bottom of the figure. Each request, indication, confirmation, response, and event notification is labeled. The meaning of each request, indication, confirmation, response, and event notification is described in the state description text.

Messages with unfilled arrowheads are passed to or from the state machine's transmitter or receiver, not shown in the state machine figures, and are directly related to data being transmitted on or received from the physical link.

The state machine description text for each state wholly defines the messages sent while the state machine is in that state. If a state is repeatedly sending a message transitions to another state, then that state stops sending that message before making the transition.

3.7.4 State machine counters, timers, and variables

State machines may contain counters, timers, and variables that affect the operation of the state machine. The following apply to counters, timers, and variables:

  a) their scope is a single state machine;
  b) they are created and deleted with the state machine or state machines with which they are associated;
  c) their initialization and modification is specified in the state descriptions and the transition descriptions; and
  d) their current values may be used to determine the behavior of a state and to select the transition out of a state.

State machine timers may continue to run while a state machine is in a given state, and a timer may cause a state transition upon reaching a defined threshold value (e.g., zero for a timer that counts down).

3.8 Bit and byte ordering

In this standard, data structures may be defined by a table. A table defines a complete ordering of elements (i.e., bits, bytes, fields, and dwords) within the structure. The ordering of elements within a table does not in itself constrain the order of storage or transmission of the data structure, but in combination with other normative text in this standard, the ordering of elements within a table may constrain the order of storage or transmission of the structure.

In a table, any element that is presented in a row above another element in a lower row is more significant than the lower element, and any element presented to the left of another element in the same row is more significant than the element to the right.

If a table shows bit numbering (see table 11), the least significant bit (LSB) is numbered 0 and each more significant bit has the next greater number than the immediately less significant bit. If a table shows numbering of bytes or characters (see table 12), the most significant byte or character is represented at the lowest number and each less significant byte or character has the next greater number than the immediately more significant byte.

In a field in a table consisting of more than one bit that contains a single value (e.g., a number), the least significant bit (LSB) is shown on the right and the most significant bit (MSB) is shown on the left (e.g., in a byte, bit 7 is the MSB and is shown on the left, bit 0 is the LSB and is shown on the right). The MSB and LSB are not labeled if the field consists of eight or fewer bits. The MSB and LSB are labeled if the field consists of more than eight bits and has no internal structure defined.

In a field in a table consisting of more than one byte that contains multiple fields each with their own values (e.g., a descriptor), there is no MSB and LSB of the field itself and thus there are no MSB and LSB labels. Each individual field has an MSB and LSB, but they are not labeled.

In a field containing a text string (e.g., ASCII or UTF-8), only the MSB of the first character and the LSB of the last character are labeled.

Multiple byte fields are represented with only two rows, with the non-sequentially increasing byte number denoting the presence of additional bytes.

A data dword consists of 32 bits. Table 11 shows a data dword containing a single value, where the MSB is on the upper left in bit 31 and the LSB is on the lower right in bit 0.

Table 11 — Example of ordering of bits and bytes within a data dword

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
2	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
3	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Note - The Bit x labels in the individual table cells are for reference only and should not appear within tables that use this element format.								

Table 12 shows a data dword containing four one-byte elements, where byte 0 (the first byte) is on the top and byte 3 (the fourth byte) is on the bottom. Each byte has an MSB on the left and an LSB on the right.

Table 12 — Example of ordering of bits and bytes within a data dword element

Bit Byte	7	6	5	4	3	2	1	0
0	First byte							
	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
1	Second byte							
	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
2	Third byte							
	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
3	Fourth byte							
	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Note - The Bit x labels in the individual table cells and the xxbyte labels in the individual bytes are for reference only and should not appear within tables that use these element formats. In this example the MSB and LSB labels are for reference only. However, they may appear in multi-byte fields as described in this subclause.								

3.9 Notation for procedure calls

In this standard, the model for functional interfaces between entities is a procedure call. Such interfaces are specified using the following notation:

[Result =] Procedure Name (IN ([input-1] [,input-2] …), OUT ([output-1] [,output-2] …))

where:

Result	A single value representing the outcome of the procedure or function.
Procedure Name	A descriptive name for the function to be performed.
IN (Input-1, Input-2, …)	A comma-separated list of names identifying caller-supplied input data objects.
OUT (Output-1, Output-2, …)	A comma-separated list of names identifying output data objects to be returned by the procedure.
[…]	Brackets enclose optional or conditional parameters and arguments.

This notation allows arguments to be specified as inputs and outputs. An interface between entities may require only inputs. If a procedure call has no output arguments, the word OUT, preceding comma, and associated pair of balanced parentheses are omitted.

The following is an example of a procedure call specification:

Found = Search (IN (Pattern, Item List), OUT ([Item Found]))

Input arguments:

Pattern: Argument containing the search pattern.
Item List: **Item<NN>** contains the items to be searched for a match.

Output arguments:

Item Found: Item located by the search procedure call. This argument is only returned if the search succeeds.

4 SCSI architecture model

4.1 Introduction

The purpose of the SCSI architecture model is to:

    a) provide a basis for the coordination of SCSI standards development that allows each standard to be placed into perspective within the overall SCSI architecture model;

    b) establish a layered model in which standards may be developed;

    c) provide a common reference for maintaining consistency among related standards; and

    d) provide the foundation for application compatibility across all SCSI interconnect and SCSI transport protocol environments by specifying generic requirements that apply uniformly to all implementation standards within each functional area.

The development of this standard is assisted by the use of a UML based abstract model. To specify the external behavior of a SCSI system, elements in a system are replaced by functionally equivalent components within this model. Only externally observable behavior is retained as the standard of behavior. The description of internal behavior in this standard is provided only to support the definition of the observable aspects of the model. Those aspects are limited to the generic properties and characteristics needed for host applications to interoperate with SCSI devices in any SCSI interconnect and SCSI transport protocol environment. The model does not address other requirements that may be essential to some I/O system implementations (e.g., the mapping from SCSI device addresses to network addresses, the procedure for discovering SCSI devices on a network, and the definition of network authentication policies for SCSI initiator devices or SCSI target devices). These considerations are outside the scope of this standard.

The set of SCSI standards specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming SCSI implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability.

The SCSI architecture model is described in terms of classes, protocol layers, and service interfaces between classes. As used in this standard, classes are abstractions, encapsulating a set of related attributes, operations, data types, and other classes. Certain classes are essential to the SCSI architecture (e.g., an interconnect), while others are needed to understand the functioning of the SCSI architecture but have implementation definitions outside the scope of the SCSI architecture (e.g., a command). These classes exhibit well-defined and observable behaviors, but they do not exist as separate physical elements. A class may contain a single attribute or be a complex entity that may:

    a) contain multiple attributes; or

    b) perform a set of operations or services on behalf of another class.

Service interfaces are defined between classes and protocol layers. The template for a distributed service interface is the client-server model described in 4.3. The structure of a SCSI I/O system is specified in 4.5 by defining the relationships among classes. The set of distributed services to be provided are described in clause 5 and clause 7.

Requirements that apply to each SCSI transport protocol standard are described in the SCSI transport protocol service model described in 5.4, 6.4, and 7.12. The model describes required behavior in terms of layers, classes within layers and SCSI transport protocol service transactions between layers.

4.2 Compliance requirements

This standard defines generic requirements that pertain to SCSI implementation standards and implementation requirements. An implementation requirement defines behavior in terms of measurable or observable parameters that apply to an implementation. Examples of implementation requirements defined in this document are the status values to be returned upon command completion and the service responses to be returned upon task management function completion.

Generic requirements are transformed to implementation requirements by an implementation standard. An example of a generic requirement is the hard reset behavior described in 6.3.2.

As shown in figure 8, all SCSI implementation standards shall reflect the generic requirements defined herein. In addition, an implementation claiming SCSI compliance shall conform to the applicable implementation requirements defined in this standard and the appropriate SCSI implementation standards. In the event of a conflict between this document and other SCSI standards under the jurisdiction of technical committee T10, the requirements of this standard shall apply.

.
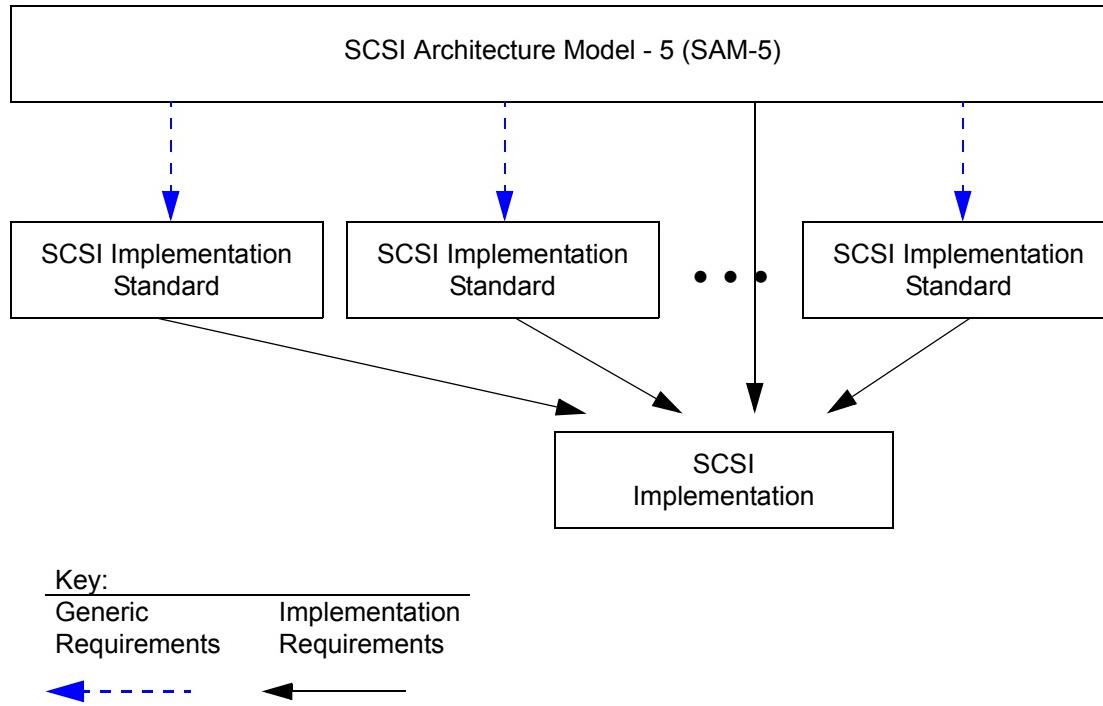
Figure 8 — Requirements precedence

4.3 The SCSI distributed service model

Service interfaces between classes are represented by the client-server model shown in figure 9. Dashed horizontal lines with arrowheads denote a single request-response transaction as it appears to the client and server. The solid lines with arrowheads indicate the actual transaction path through a service delivery subsystem. In this model, each client or server is a single thread of processing that runs concurrently with all other clients or servers.
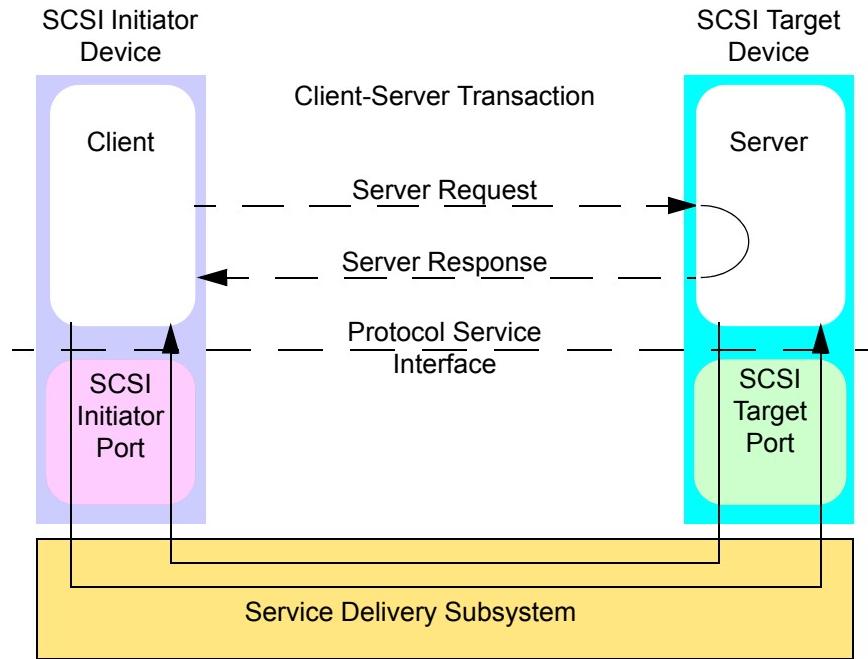
Figure 9 — Client-server model

A client-server transaction is represented as a procedure call with:

    a)   inputs supplied by the caller (i.e., the client); and
    b)   outputs, if any, and a procedure call status supplied by the server being called.

The procedure call is processed by the server and returns outputs and a procedure call status. Using the SCSI initiator port and service delivery subsystem, a client:

    a)   sends server requests to a remote server; and
    b)   receives a server response or a failure notification.

The server request identifies the server, the service to be performed, and includes the input data. A server response conveys the output data and server request status. A failure notification indicates that a condition has been detected (e.g., a reset or service delivery failure) that prevents completion of the server request.

As seen by the client, a server request becomes pending when it is passed to the SCSI initiator port for transmission and completes when the server response or failure notification is transferred to the client by the SCSI initiator port. As seen by the server, the server request becomes pending upon receipt from the SCSI target port and completes when the server response is passed to the SCSI target port for return to the client. As a result there may be a time skew between the server's and client's perception of server request status and server condition.

Client-server relationships are asymmetric. A client only originates server requests. A server only responds to such server requests.

The client sends a server request to a server located in another SCSI device and waits for completion that includes transmission of the server request to the remote server and transmission of the server response from the remote server. From the client's point of view, the behavior of a service requested from and processed in a different SCSI device than the client.is indistinguishable from a server request from and processed in the same SCSI device as the client. The client is not required to confirm successful delivery of server request or server response. Confirming successful delivery and detection of delivery failures is performed by the SCSI initiator port in cooperation with the service delivery subsystem.

4.4 The SCSI client-server model

4.4.1 SCSI client-server model overview

As shown in figure 10, each SCSI target device provides services performed by device servers and task management functions performed by task managers. A logical unit is a class that implements one of the device functional models described in the SCSI command standards and processes commands (e.g., reading from or writing to the media). Each command defines a unit of work to be performed by the logical unit that may be externally referenced and controlled through requests issued to the task manager.
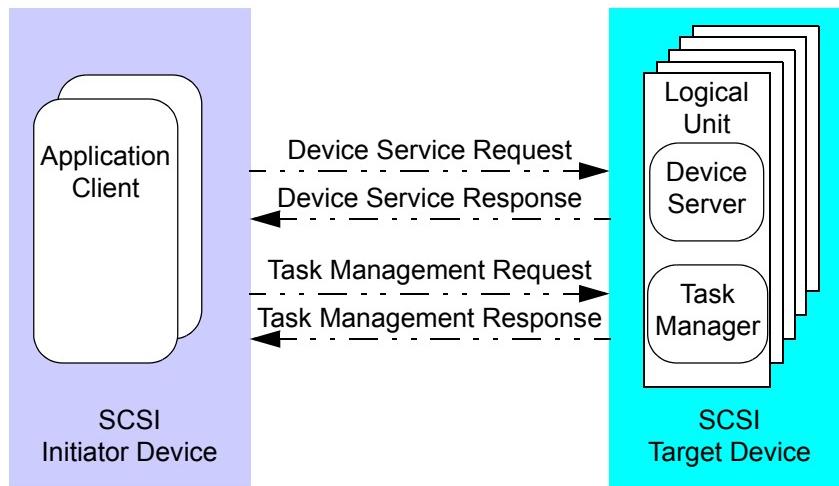
Figure 10 — SCSI client-server model

All requests originate from application clients residing within a SCSI initiator device. An application client is independent of the interconnect and SCSI transport protocol (e.g., an application client may communicate to the device driver and any other code within the operating system that is capable of managing I/O requests without requiring knowledge of the interconnect or SCSI transport protocol).

As described in 4.3, each request takes the form of a procedure call with arguments and a status to be returned. An application client may request processing of a command or a task management function through a request directed to the device server within a logical unit. The application client request is received by the addressed logical unit's task manager and is processed by:

   a)   that task manager if the request is a task management function (see clause 7); or
   b)   the device server if the request is a command (see clause 5).

4.4.2 Synchronizing client and server changes

One way a client (e.g., application client) is informed of server (e.g., device server) changes is through the arrival of server responses. The client notification of server changes occurs after the server has sent the associated server response (e.g., the SCSI target device changes before invoking the **Send Command Complete** SCSI transport protocol service response (see 5.4.2.4), but the application client is not informed of the change until the **Command Complete Received** SCSI transport protocol service confirmation (see 5.4.2.5) arrives) The client notification of server changes also occurs after the server response has been received by the SCSI initiator device.

SCSI transport protocols may require the SCSI target device to verify that the server response has been received without error before completing a change(i.e., a synchronized change). Since synchronized changes are not assumed or required by the SCSI architecture model, there may be a time lag between the occurrence of a change within the SCSI target device and the SCSI initiator device's awareness of that change.

This standard assumes that synchronized changes, if required by a SCSI transport protocol standard, are enforced by a service delivery subsystem transparently to the server (i.e., whenever the server invokes a SCSI transport protocol service to return a response as described in 7.12 and 5.4, the SCSI architecture model requires that the SCSI port for such a SCSI transport protocol does not return control to the server until the server response has been delivered without error to the SCSI initiator device).

4.4.3 Server request/response ordering

Server request transactions or server response transactions are in order if, relative to a given pair of sending and receiving SCSI ports, transactions are delivered in the order they were sent.

A sender may require control over the order in which its server requests or server responses are delivered to the receiver (e.g., the sequence in which server requests are received is important whenever a SCSI initiator device issues a series of commands with the ORDERED task attribute to a logical unit as described in clause 8). In this case, the order in which these commands are completed, and hence the final condition of the logical unit, may depend on the order in which these commands are received. The SCSI initiator device may develop knowledge about the condition of commands and task management functions and may take action based on the nature and sequence of SCSI target device responses (e.g., a SCSI initiator device should be aware that further responses are possible from an aborted command because the command completion response may be delivered out of order with respect to the response to the abort request).

The manner in which ordering constraints are established is vendor specific. An implementation may delegate this requirement to the application client (e.g., the device driver). In-order delivery may be an intrinsic property of a service delivery subsystem or a requirement established by the SCSI transport protocol standard.

The order in which task management requests are processed is not specified by the SCSI architecture model. The SCSI architecture model does not require in-order delivery of task management requests or processing by the task manager in the order received. To guarantee the processing order of task management requests sent to a specific logical unit, an application client should not have more than one such task management request pending to that logical unit.

To simplify the description of behavior, the SCSI architecture model assumes in-order delivery of server requests or server responses to be a property of a service delivery subsystem. This assumption does not constitute a requirement. The SCSI architecture model makes no assumption about and places no requirement on the ordering of server requests or server responses for different I_T nexuses.

4.5 The SCSI structural model

The SCSI structural model represents a view of the classes in a SCSI I/O system as seen by the application clients interacting with that SCSI I/O system. As shown in figure 11, the SCSI I/O system is represented by a SCSI Domain class. A SCSI domain contains SCSI devices (i.e. instances of the SCSI Device class) and a service delivery subsystem (i.e. an instance of the Service Delivery Subsystem class) that transports commands, data, task management functions, and related information. A SCSI device contains clients, servers (see 4.4), or both and the infrastructure to support them.
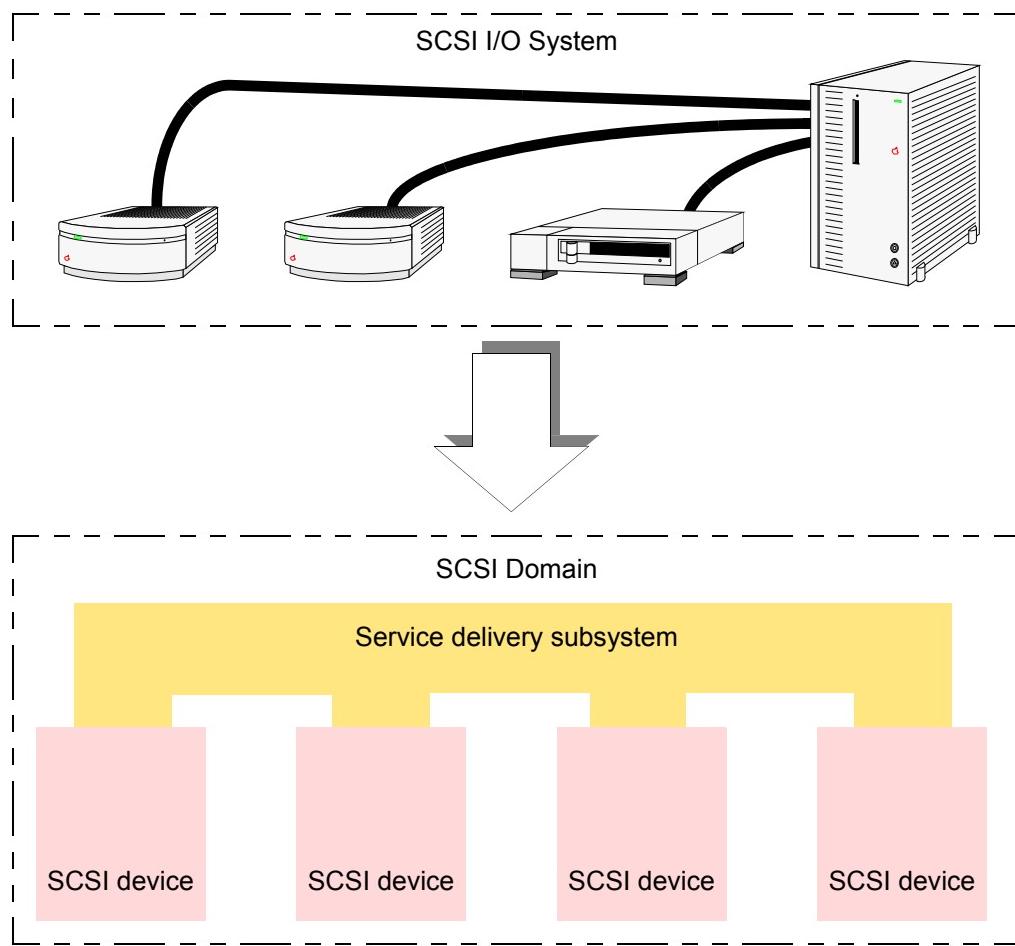
Figure 11 — SCSI I/O system and SCSI domain model

4.6 SCSI classes

4.6.1 SCSI classes overview

Figure 12 shows the main classes of the SCSI domain. This standard defines these classes in greater detail.
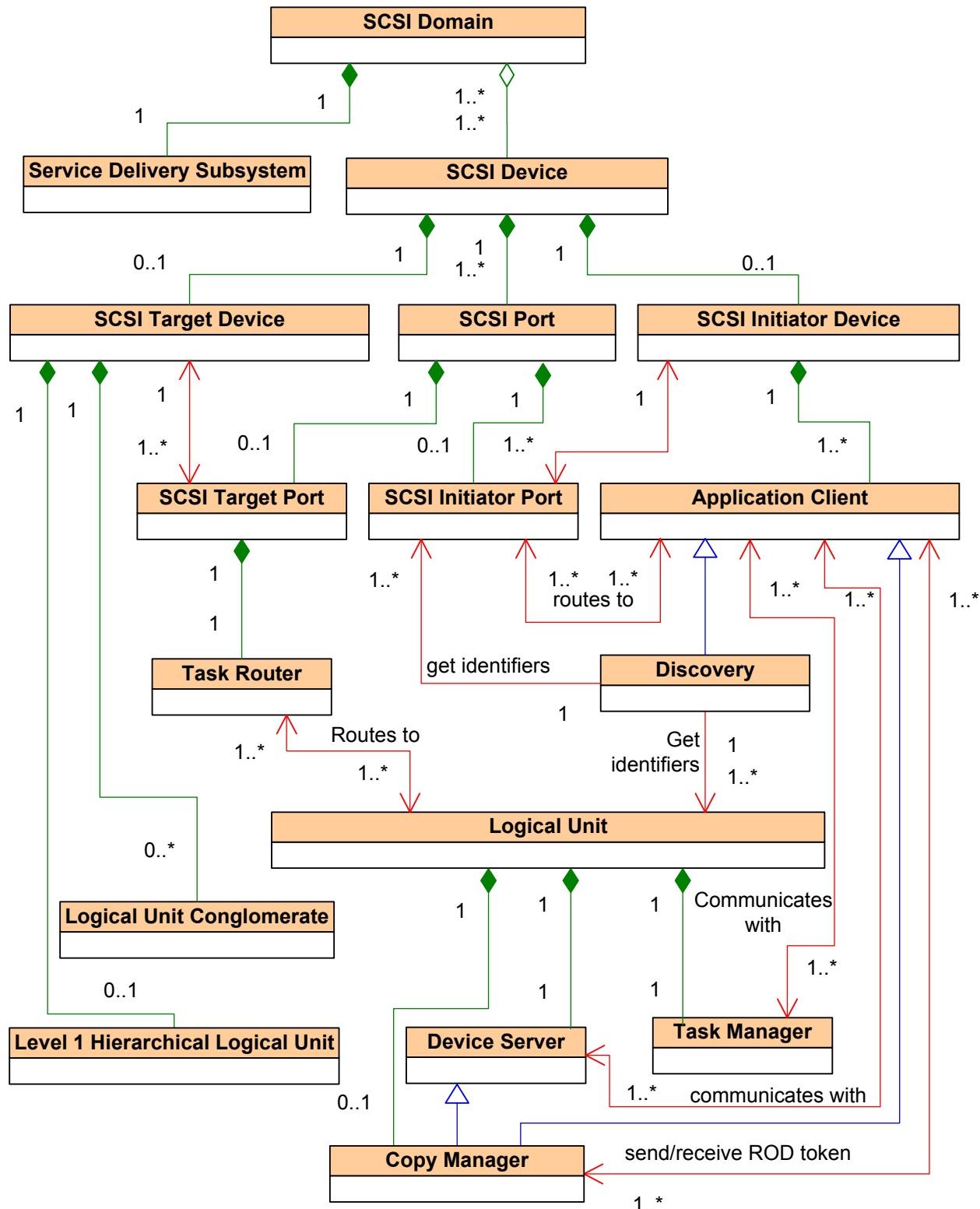
Figure 12 — SCSI Domain class diagram overview

4.6.2 SCSI Domain class

The SCSI Domain class (see figure 13) contains the:

    a) Service Delivery Subsystem class (see 4.6.3); and
    b) SCSI Device class (see 4.6.4).
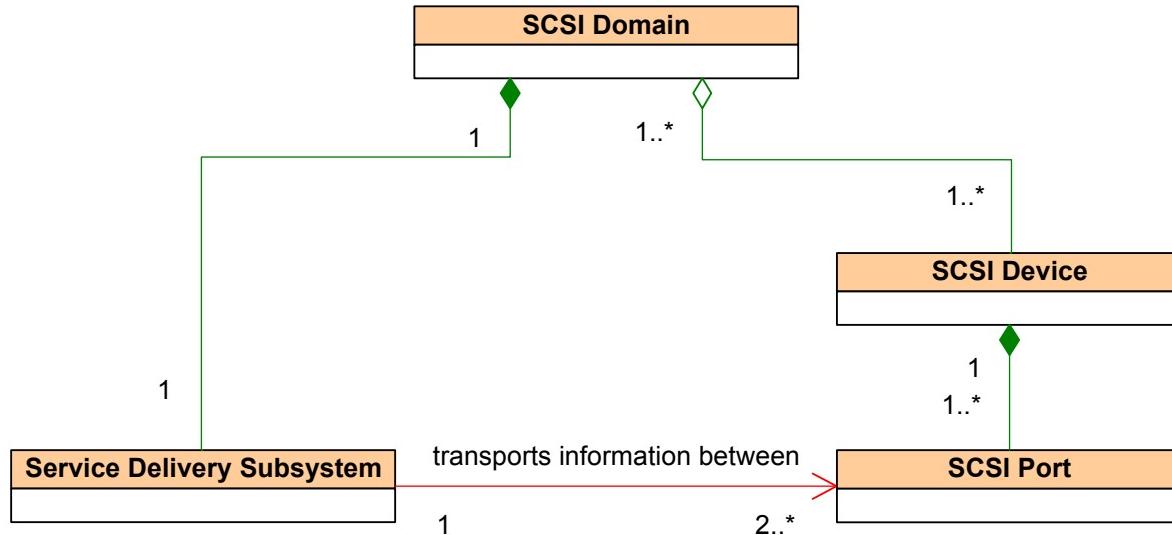
Figure 13 — SCSI Domain class diagram

Each instance of a SCSI Domain class shall contain the following objects:

    a) one service delivery subsystem; and
    b) one or more SCSI devices, each of which shall contain:
        A) one or more SCSI ports.

Instantiation requirements for the SCSI Device class are also described in 4.6.4.1.

Instantiation requirements for the SCSI Port class are also described in 4.6.5.1.

See figure 14 for the instantiation that contains the minimum set of objects that make up a valid SCSI domain.
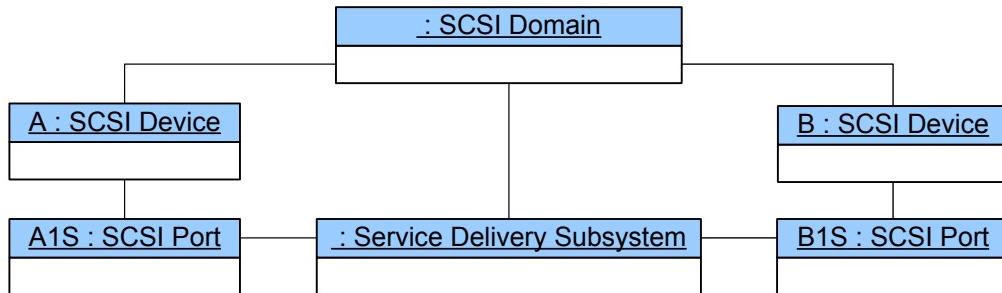
Figure 14 — SCSI Domain object diagram

The boundaries of a SCSI domain are established by the SCSI I/O system (see 4.5) implementor, within the constraints of a specific SCSI transport protocol and associated interconnect standards.

4.6.3 Service Delivery Subsystem class

The Service Delivery Subsystem class (see figure 13) connects all the SCSI ports in the SCSI domain, providing a mechanism through which clients communicate with servers.

A service delivery subsystem is composed of one or more interconnects that appear to a client or server as a single path for the transfer of requests and responses between SCSI devices.

A service delivery subsystem is assumed to provide error-free transmission of requests and responses between client and server. Although a client (e.g., device driver in a SCSI implementation) may perform these transfers through several interactions with its STPL, the SCSI architecture model portrays each operation, from the viewpoint of the client, as occurring in one discrete step. The request or response is considered:

    a) sent by the sender when the sender passes it to the SCSI port for transmission;
    b) in transit until delivered; and
    c) received by the receiver when the server request or server response has been forwarded to the receiver via the destination SCSI device's SCSI port.

4.6.4 SCSI Device class

4.6.4.1 SCSI Device class overview

The SCSI Device class (see figure 15) contains:

    a) the SCSI Port class (see 4.6.5); and
    b) the SCSI Initiator Device class (see 4.6.27), the SCSI Target Device class (see 4.6.9), or both.
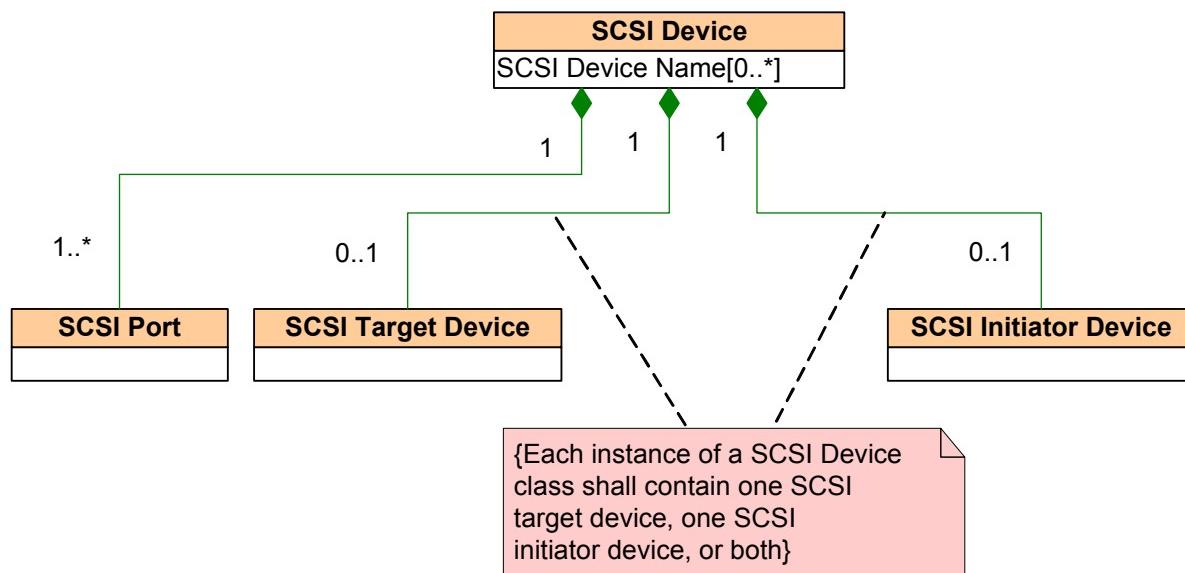
Figure 15 — SCSI Device class diagram

Each instance of a SCSI Device class shall contain:

    a) one or more SCSI ports; and
    b) one SCSI target device, one SCSI initiator device, or both.

Instantiation requirements for the SCSI Port class are also described in 4.6.5.1.

Instantiation requirements for the SCSI Target Device class are also described in 4.6.9.

Instantiation requirements for the SCSI Initiator Device class are also described in 4.6.27.

4.6.4.2 SCSI Device Name attribute

The SCSI Device Name attribute contains one or more names for a SCSI device that is world wide unique within the SCSI transport protocol of each SCSI domain in which the SCSI device has SCSI ports. For each supported SCSI transport protocol, a SCSI device shall have no more than one (i.e., zero or one) SCSI Device Name attribute that is not in the SCSI name string format (see SPC-4). A SCSI device shall have no more than one (i.e., zero or one) SCSI Device Name attribute in the SCSI name string format regardless of the number of SCSI transport protocols supported by the SCSI device. If a SCSI device has a SCSI Device Name attribute in the SCSI name string format then the SCSI device should have only one SCSI Device Name attribute. A SCSI device name shall never change and may be used for persistent identification of a SCSI device in contexts where specific references to initiator port names, target port names, initiator port identifiers, or target port identifiers are not required.

A SCSI transport protocol standard may require that a SCSI device include a SCSI Device Name attribute if the SCSI device has SCSI ports in a SCSI domain of that SCSI transport protocol. The SCSI Device Name attribute may be made available to other SCSI devices or SCSI ports in a given SCSI domain in SCSI transport protocol specific ways.

A SCSI device name for a SCSI target device may be reported in a Device Identification VPD page's designation descriptor for the SCSI target device (see SPC-4). A SCSI device name for a SCSI initiator device is reported by methods outside the scope of this standard.

4.6.5 SCSI Port class

4.6.5.1 SCSI Port class overview

The SCSI Port class (see figure 16) contains:

   a)   the SCSI Target Port class (see 4.6.6);
   b)   the SCSI Initiator Port class (see 4.6.8); or
   c)   both.
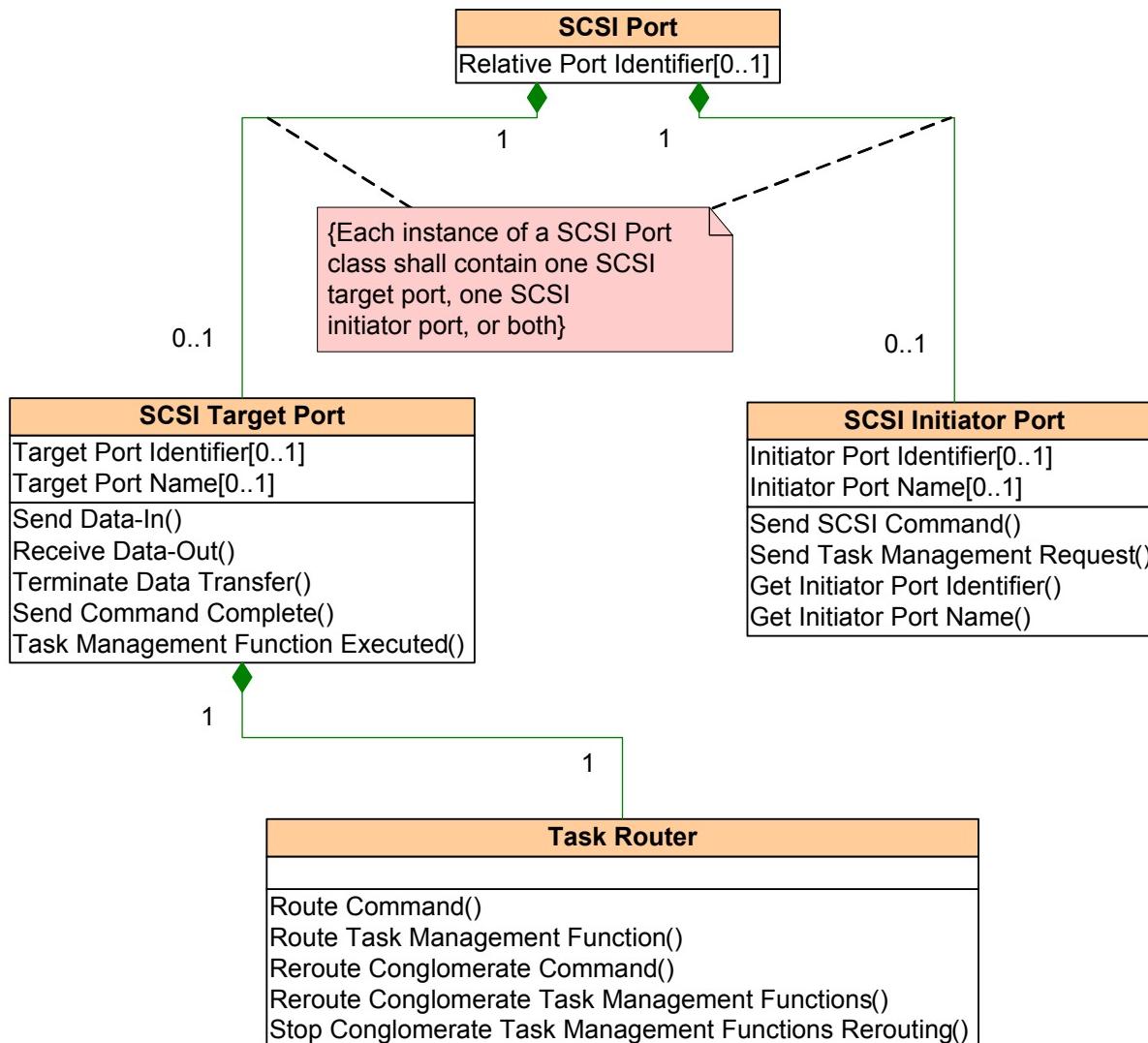
Figure 16 — SCSI Port class diagram

Each instance of a SCSI Port class shall contain:

   a)   one SCSI target port that shall contain:
        A)   one task router;
   b)   one SCSI initiator port; or
   c)   both.

4.6.5.2 Relative Port Identifier attribute

The Relative Port Identifier attribute identifies a SCSI target port or a SCSI initiator port relative to other SCSI ports in a SCSI target device and any SCSI initiator devices contained within that SCSI target device. A SCSI target device may assign relative port identifiers to its SCSI target ports and any SCSI initiator ports. If relative port identifiers are assigned, the SCSI target device shall assign each of its SCSI target ports and any SCSI initiator ports a unique relative port identifier from 1 to 65 535. SCSI target ports and SCSI initiator ports share the same number space.

Examples of relative port identifiers usage are as described in the Device Identification VPD page (see SPC-4) and the SCSI Ports VPD page (see SPC-4).

The relative port identifiers are not required to be contiguous. The relative port identifier for a SCSI port shall not change once assigned unless reconfiguration of the SCSI target device occurs.

4.6.6 SCSI Target Port class

4.6.6.1 SCSI Target Port class overview

The SCSI Target Port class (see figure 16) contains:

    a) the Task Router class (see 4.6.7).

Instantiation requirements for the SCSI Target Port class are described in 4.6.5.1.

The SCSI Target Port class connects SCSI target devices to a service delivery subsystem.

The SCSI Target Port class is associated with the SCSI Target Device class (see figure 12).

The SCSI target port invokes:

    a) the Data-In Delivered operation (see 5.4.3.2.2) of a device server after sending data over the service delivery subsystem;
    b) the Data-Out Received operation (see 5.4.3.3.2) of a device server after receiving data over the service delivery subsystem;
    c) the Data Transfer Terminated operation (see 5.4.3.4.2) of a device server or a task manager after terminating data transfers;
    d) the Route Command operation (see 4.6.7.2) of the task router in response to receiving a command over the service delivery subsystem;
    e) the Route Task Management Function operation (see 4.6.7.3) of the task router in response to receiving a task management function over the service delivery subsystem;
    f) the Nexus Loss operation (see 4.6.21.5) of a task manager in response to an I_T nexus loss (see 6.3.4);
    g) the Transport Reset operation (see 4.6.21.6) of a task manager in response to a hard reset (see 6.3.2); and
    h) the Power Loss Expected operation (see 4.6.21.7) of a task manager in response to a power loss expected (see 6.3.5).

4.6.6.2 Target Port Identifier attribute

The Target Port Identifier attribute, if any, contains a target port identifier for a SCSI target port. The target port identifier is a value by which a SCSI target port is referenced within a SCSI domain.

If a transport protocol standard supports more than one SCSI target port within a SCSI domain, then that transport protocol standard shall define target port identifiers. If a transport protocol standard only supports one SCSI target port within a SCSI domain, then that transport protocol standard is not required to define target port identifiers.

The target port identifier may be reported in a target port name designation descriptor in the Device Identification VPD page (see SPC-4). If a SCSI target port has a target port identifier and a target port name see SPC-4 to determine which is reported.

4.6.6.3 Target Port Name attribute

A Target Port Name attribute, if any, contains an optional name of a SCSI target port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI target port. A SCSI target port shall have at most one name. A target port name shall never change and may be used for persistent identification of a SCSI target port.

A SCSI transport protocol standard may require that a SCSI target port include a SCSI target port name if the SCSI target port is in a SCSI domain of that SCSI transport protocol. The target port name may be made available to other SCSI devices or SCSI ports in the given SCSI domain in SCSI transport protocol specific ways.

The target port name may be reported in a target port name designation descriptor in the Device Identification VPD page (see SPC-4). If a SCSI target port has a target port identifier and a target port name, see SPC-4 to determine which is reported.

4.6.6.4 Send Data-In operation

The Send Data-In operation implements the Send Data-In SCSI transport protocol service request (see 5.4.3.2.1) by sending data over the service delivery subsystem.

4.6.6.5 Receive Data-Out operation

The Receive Data-Out operation implements the Receive Data-Out SCSI transport protocol service request (see 5.4.3.3.1) by receiving data over the service delivery subsystem.

4.6.6.6 Terminate Data Transfer operation

The Terminate Data Transfer operation implements the Terminate Data Transfer SCSI transport protocol service request (see 5.4.3.4.1) by terminating data transfers.

4.6.6.7 Send Command Complete operation

The Send Command Complete operation implements the Send Command Complete SCSI transport protocol service response (see 5.4.2.4) by transmitting command complete information over the service delivery subsystem.

4.6.6.8 Task Management Function Executed operation

The Task Management Function Executed operation implements the Task Management Function Executed SCSI transport protocol service response (see 7.12.4) by transmitting task management function executed information over the service delivery subsystem.

4.6.7 Task Router class

4.6.7.1 Task Router class overview

The Task Router class (see figure 16) routes commands and task management functions.

In some transport protocols, the task router may check for overlapped command identifiers on commands (see 5.10).

The task router invokes:

   a) the Terminate Data Transfer operation (see 4.6.6.6) of a SCSI target port to terminate data transfers for a command that it is aborting; and
   b) the Send Command Complete operation ((see 4.6.6.7) of a SCSI target port to complete processing a command.

4.6.7.2 Route Command operation

This operation is modeled by the following procedure call:

Route Command   (IN (I_T_L Nexus, Command Identifier, CDB, Task Attribute, [CRN], [Command Priority], [First Burst Enabled]))

Input arguments:

I_T_L Nexus: An identifier for the I_T_L nexus routing the command (see 4.6.32.3).

Command Identifier: The numerical identifier identifying the command (see 4.6.31.3).

CDB: Command descriptor block (see 5.2).

Task Attribute: A value specifying one of the task attributes defined in 8.4. For specific requirements on the Task Attribute argument see 8.4.

CRN: If a CRN argument is used, then all commands on an I_T_L nexus include a CRN argument (see 5.1).

Command Priority: The priority assigned to the command (see 8.5).

First Burst Enabled: An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer are being delivered to the logical unit without waiting for the device server to invoke the Receive Data-Out SCSI transport protocol service.

The Route Command operation routes a command to a task manager as follows:

1) if the requirements specified by the Reroute Conglomerate Command operation (see 4.6.7.4) are met, then a command associated with a valid subsidiary logical unit is rerouted to the task manager in the administrative logical unit associated with that subsidiary logical unit;
2) commands associated with a valid logical unit are routed to the task manager in the specified logical unit by invoking the SCSI Command Received operation (see 4.6.21.2) of that task manager;
3) commands that are associated with an incorrect logical unit number in which the ADMINISTRATIVE ELEMENT field (see 4.7.6.2) specifies an administrative logical unit (see 4.6.11) are routed to the task manager in that administrative logical unit by invoking the SCSI Command Received operation of the task manager in the logical unit that is specified by setting the subsidiary element to zero (i.e., using the algorithm described in 4.6.10) without modifying the contents of the I_T_L nexus or the command identifier to the SCSI Command Received operation; and
4) commands associated with an incorrect logical unit number are handled as described in 5.11.

4.6.7.3 Route Task Management Function operation

This operation is modeled by the following procedure call:

Route Task Management Function   (IN (Nexus, [Command Identifier], Function Identifier))

Input arguments:

Nexus: An identifier for the I_T nexus (see 4.6.32.2) or, identifier for the I_T_L nexus (see 4.6.32.3).

Command Identifier: The numerical identifier identifying the command (see 4.6.31.3).

Function Identifier: Argument encoding the task management function to be performed.

The Route Task Management Function operation routes a task management function to one or more task managers as follows:

1) if the requirements specified by the Reroute Conglomerate Task Management Functions operation (see 4.6.7.5) are met, then task management functions associated with a valid subsidiary logical unit may be rerouted to the task manager in the administrative logical unit associated with that subsidiary logical unit;
2) task management functions with a scope of:

     A) I_T_L nexus (e.g., ABORT TASK SET, CLEAR TASK SET, CLEAR ACA, LOGICAL UNIT RESET, QUERY TASK SET, and QUERY ASYNCHRONOUS EVENT); or

     B) I_T_L nexus and command identifier (e.g., ABORT TASK and QUERY TASK) associated with a valid logical unit,

are routed to the task manager in the specified logical unit by invoking the Task Management Request Received operation (see 4.6.21.3) of that task manager;

   3) task management functions with a scope of I_T nexus (e.g., I_T NEXUS RESET) are routed to the task manager in each logical unit by invoking the Task Management Request Received operation of each of those task managers; and

   4) task management functions with a scope of:

     A) I_T_L nexus; or

     B) I_T_L nexus and command identifier,

associated with an incorrect logical unit are handled as described in 7.12.

4.6.7.4 Reroute Conglomerate Command operation

This operation is modeled by the following procedure call:

Reroute Conglomerate Command   (IN (LUN))

Input arguments:

     LUN: The LUN for the administrative logical unit in a logical unit conglomerate (see 4.6.10).

The processing of a Reroute Conglomerate Command operation causes the task router to reroute to the specified administrative logical unit (see 4.6.11) the next command received that is associated with any valid subsidiary logical unit (see 4.6.26) in the logical unit conglomerate whose administrative logical unit is specified by the LUN argument.

The processing of a Reroute Conglomerate Command operation causes the Route Command operation (see 4.6.7.2) to reroute the next received command using the following method:

   1) the I_T_L nexus and the command identifier of every command processed by the Route Command operation is monitored for a command that is associated with one of the valid subsidiary logical units in the logical unit conglomerate whose administrative logical unit is specified by the LUN argument to the Reroute Conglomerate Command operation using the method described in 4.6.10 to detect the LUNs of valid subsidiary logical units;

   2) the first command associated with a specified valid subsidiary logical unit is rerouted to the specified administrative logical unit by invoking the SCSI Command Received operation (see 4.6.21.2) of the administrative logical unit's task manager without modifying the contents of the I_T_L nexus or the command identifier to the Route Command operation; and

   3) after a command is rerouted, no other commands are rerouted until a Reroute Conglomerate Command operation is processed.

The processing of a logical unit reset event (see 6.3.3) causes the task router to discard the task rerouting information stored for any Reroute Conglomerate Command operations that have not yet resulted in a command being rerouted.

Commands that are associated with an incorrect logical unit number (see 5.11) are not affected by the Reroute Conglomerate Command operation.

4.6.7.5 Reroute Conglomerate Task Management Functions operation

This operation is modeled by the following procedure call:

Reroute Conglomerate Task Management Functions   (IN (LUN))

Input arguments:

> **LUN:** The LUN for the administrative logical unit in a logical unit conglomerate (see 4.6.10).

Until stopped, as described in this subclause, the processing of a Reroute Conglomerate Task Management Functions operation causes the task router to reroute to the administrative logical unit (see 4.6.11) all task management functions received that are associated with any valid subsidiary logical unit (see 4.6.26) in the logical unit conglomerate whose administrative logical unit is specified by the LUN argument.

The processing of a Reroute Conglomerate Task Management Functions operation causes the Route Task Management Function operation (see 4.6.7.3) to reroute received task management functions using the following method:

1) the I_T_L nexus and the command identifier, if any, of every task management function processed by the Route Task Management Function operation is monitored for a task management function that is associated with one of the valid subsidiary logical units in the logical unit conglomerate whose administrative logical unit is specified by the LUN argument to the Reroute Conglomerate Task Management Function operation using the method described in 4.6.10 to detect the LUNs of valid subsidiary logical units; and
2) any task management function associated with a specified valid subsidiary logical unit is rerouted to the specified administrative logical unit by invoking the Task Management Request Received operation (see 4.6.21.3) of that administrative logical unit's task manager without modifying the contents of the I_T_L nexus or the command identifier, if any, to the Route Task Management Function operation.

The rerouting of task management functions is stopped by the processing of a:

a) Stop Conglomerate Task Management Functions Rerouting operation (see 4.6.7.6); or
b) logical unit reset event (see 6.3.3).

Task management functions that are associated with an incorrect logical unit number (see 5.11) are not affected by the Reroute Conglomerate Task Management Functions operation.

4.6.7.6 Stop Conglomerate Task Management Functions Rerouting operation

This operation is modeled by the following procedure call:

> **Stop Conglomerate Task Management Functions Rerouting (IN (LUN))**

Input arguments:

> **LUN:** The LUN for the administrative logical unit in a logical unit conglomerate (see 4.6.10).

The processing of a Stop Conglomerate Task Management Functions Rerouting operation ends the rerouting of task management functions:

a) that are associated with any valid subsidiary logical unit (see 4.6.26) in the logical unit conglomerate whose administrative logical unit (see 4.6.11) is specified by the LUN argument; and
b) for which rerouting was begun by a prior Reroute Conglomerate Task Management Functions operation (see 4.6.7.5).

4.6.8 SCSI Initiator Port class

4.6.8.1 SCSI Initiator Port class overview

The SCSI Initiator Port class (see figure 16) connects SCSI initiator devices to a service delivery subsystem.

The SCSI Initiator Port class is associated with the SCSI Initiator Device class (see figure 12).

The SCSI initiator port invokes:

   a) the Command Complete Received operation (see 4.6.28.2) of an application client after receiving command complete information over the service delivery subsystem;
   b) the Received Task Management Function Executed operation (see 4.6.28.3) of an application client after receiving task management function executed information over the service delivery subsystem;
   c) the Nexus Loss operation (see 4.6.28.4) of an application client in response to an I_T nexus loss (see 6.3.4); and
   d)   the Transport Reset operation (see 4.6.28.5) of an application client in response to a hard reset (see 6.3.2).

4.6.8.2 Initiator Port Identifier attribute

The Initiator Port Identifier attribute, if any, contains an initiator port identifier for a SCSI initiator port. The initiator port identifier is a value by which a SCSI initiator port is referenced within a SCSI domain.

If a transport protocol standard supports more than one SCSI initiator port within a SCSI domain, then that transport protocol standard shall define initiator port identifiers. If a transport protocol standard only supports one SCSI initiator port within a SCSI domain, then that transport protocol standard is not required to define initiator port identifiers.

The initiator port identifier is reported by methods outside the scope of this standard.

4.6.8.3 Initiator Port Name attribute

A Initiator Port Name attribute, if any, contains an optional name of a SCSI initiator port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI initiator port. A SCSI initiator port shall have at most one name. An initiator port name shall never change and may be used for persistent identification of a SCSI initiator port.

A SCSI transport protocol standard may require that a SCSI initiator port include a SCSI initiator port name if the SCSI initiator port is in a SCSI domain of that SCSI transport protocol. The initiator port name may be made available to other SCSI devices or SCSI ports in the given SCSI domain in SCSI transport protocol specific ways.

The initiator port name is reported by methods outside the scope of this standard.

4.6.8.4 Send SCSI Command operation

The Send SCSI Command operation implements the Send SCSI Command SCSI transport protocol service request (see 5.4.2.2) by sending a command over the service delivery subsystem.

4.6.8.5 Send Task Management Request operation

The Send Task Management Request operation implements the Send Task Management Request SCSI transport protocol service request (see 7.12.2) by sending a task management function over the service delivery subsystem.

4.6.8.6 Get Initiator Port Identifier operation

The Get Initiator Port Identifier operation returns the Initiator Port Identifier attribute, if any, of the SCSI initiator port to the requesting Discovery class.

4.6.8.7 Get Initiator Port Name operation

The Get Initiator Port Name operation returns the Initiator Port Name attribute, if any, of the SCSI initiator port to the requesting Discovery class.

4.6.9 SCSI Target Device class

The SCSI Target Device class (see figure 17) is a SCSI Device class that contains:

    a)   the Level 1 Hierarchical Logical Unit class (see 4.6.11); and
    b)   the Logical Unit Conglomerate class (see 4.6.10).
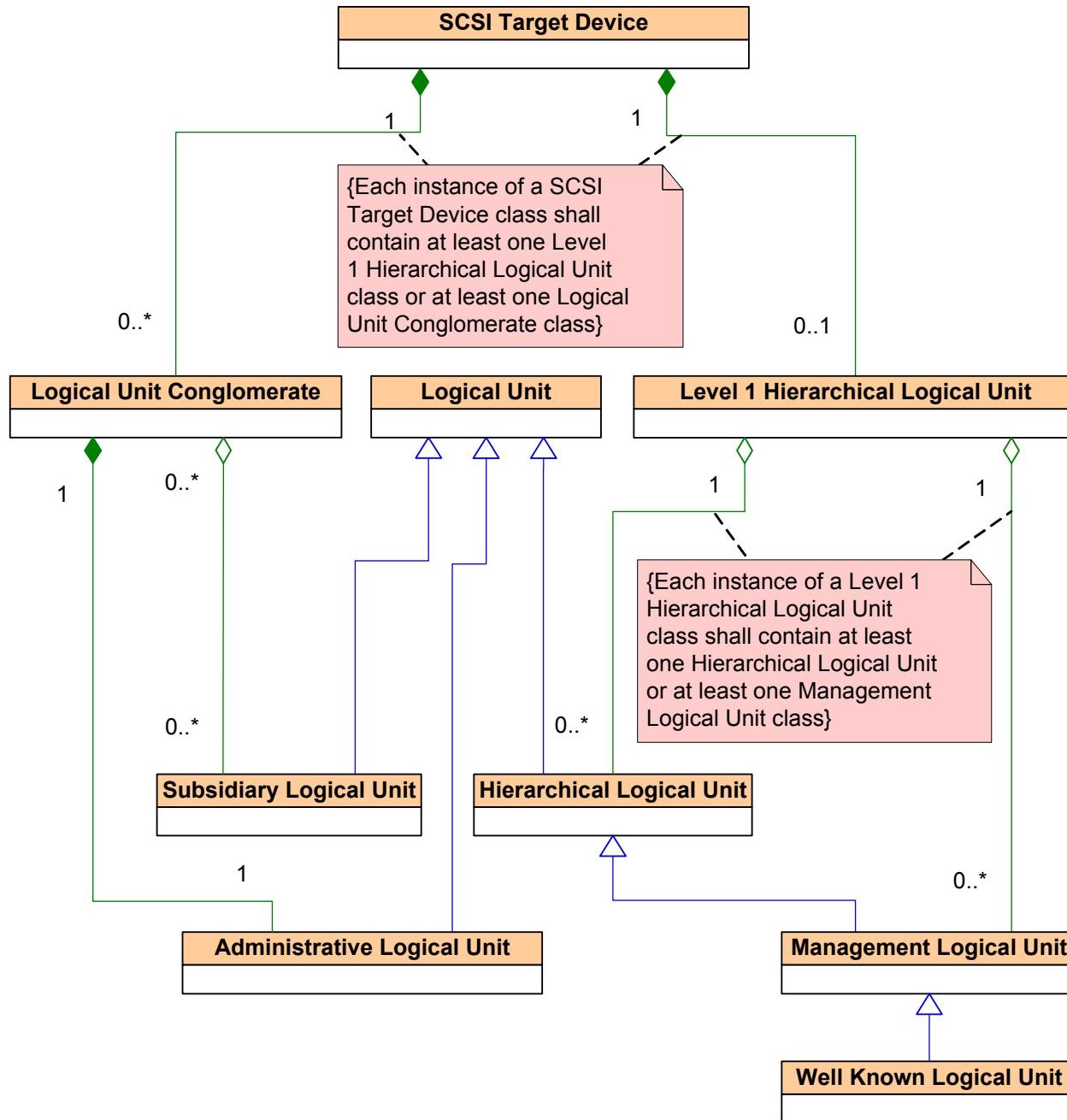
Figure 17 — SCSI Target Device class diagram

Each instance of the SCSI Target Device class shall contain the following objects:

    a)   zero or one level 1 hierarchical logical unit that contains:
        A)   at least one hierarchical logical unit or management logical unit;
        B)   zero or more hierarchical logical units;
        C)   zero or more management logical units; and

D) zero or more well known logical units;

and

b) zero or more logical unit conglomerates that contain:
   A) one administrative logical unit; and
   B) zero or more subsidiary logical units.

Each instance of the SCSI Target Device class shall contain at least one level 1 hierarchical logical unit or logical unit conglomerate.

Instantiation requirements for the Level 1 Hierarchical Logical Unit class are also described in 4.6.11.

Instantiation requirements for the Logical Unit class are described in 4.6.18.

The SCSI Target Device class is associated with the SCSI Target Port class (see figure 12).

4.6.10 Logical Unit Conglomerate class

The Logical Unit Conglomerate class (see figure 17) contains:

a) the Administrative Logical Unit class (see 4.6.11); and
b) the Subsidiary Logical Unit class (see 4.6.26).

Instantiation requirements for the Logical Unit Conglomerate class are described in 4.6.9.

Administrative functionality (see 4.6.11) provided by the device server in the administrative logical unit may:

a) affect subsidiary logical units, if any, in the logical unit conglomerate when an administrative function is performed by the administrative logical unit's device server; and
b) represent the administrative logical unit and subsidiary logical units, if any, in the logical unit conglomerate when a condition is established that affects information returned to the application client (e.g., unit attention coalescing (see 5.14.2)).

Administrative functionality that affects the logical unit conglomerate is requested from outside the SCSI target device (e.g., by an application client). Administrative functionality that represents the logical unit conglomerate may cause information (e.g., asymmetric access state) to be transferred outside the SCSI target device (e.g., to an application client).

The conditions that cause administrative functionality to be performed include:

a) requests (i.e., commands) sent by application clients; and
b) events and requests that are outside the scope of this standard (e.g., requests received from outside the SCSI domain).

The administrative logical unit and all subsidiary logical units, if any, in a logical unit conglomerate shall:

a) use the logical unit conglomerate LUN structure (see 4.7.6.2); and
b) set the LU_CONG bit to one in their standard INQUIRY data (see SPC-4).

The logical unit conglomerate LUN structure (see 4.7.6.2) is constructed to allow the determination of the LUN of the administrative logical unit in a logical unit conglomerate based on the LUN for a subsidiary logical unit in the same logical unit conglomerate. If the bits that specify the subsidiary logical unit in its LUN (i.e., the bits in the SUBSIDIARY ELEMENT field) are set to zero, then the resulting LUN is that of the administrative logical unit in the same logical unit conglomerate.

For each subsidiary logical unit within a logical unit conglomerate, the SUBSIDIARY ELEMENT field (see 4.7.6.2) in each LUN, if any, shall be:

a) unique to that subsidiary logical unit; and
b) identical for all LUNs for that subsidiary logical unit.

4.6.11 Level 1 Hierarchical Logical Unit class

The Level 1 Hierarchical Logical Unit class (see figure 17 and figure 18) contains:

 a) the Hierarchical Logical Unit class (see 4.6.15);
 b) the Management Logical Unit class (see 4.6.16);
 c) the Well Known Logical Unit class (see 4.6.17); and
 d) the Level 2 Hierarchical Logical Unit class (see 4.6.12).
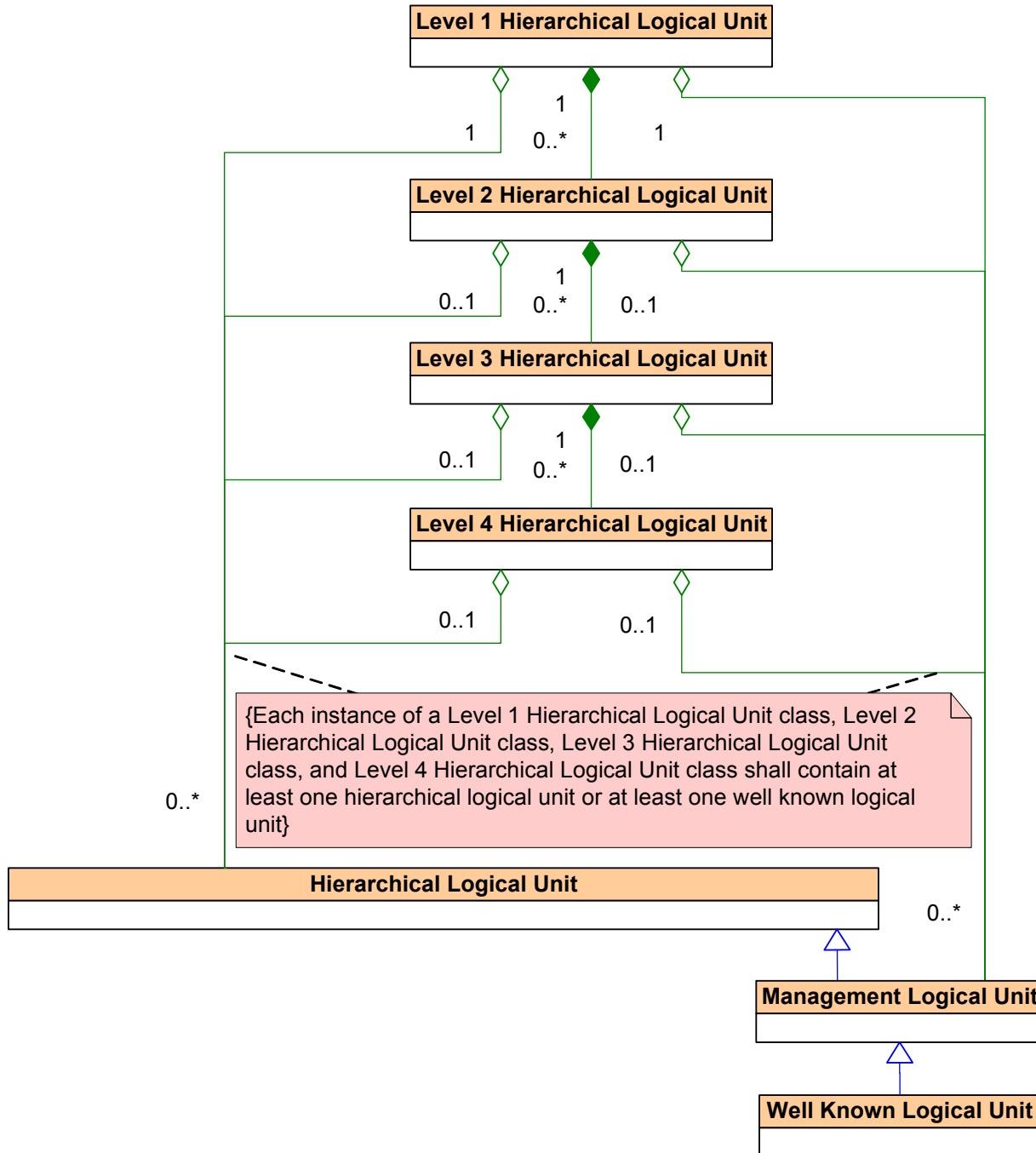
Figure 18 — Level 1 Hierarchical Logical Unit class

Each instance of a Level 1 Hierarchical Logical Unit class shall contain the following objects:

a)   at least one hierarchical logical unit or management logical unit;
b)   zero or more hierarchical logical units;
c)   zero or more management logical units;
d)   zero or more well known logical units;
e)   zero or more level 2 hierarchical logical units;
f)   zero or more level 3 hierarchical logical units; and
g)   zero or more level 4 hierarchical logical units.

Each instance of a Level 2 Hierarchical Logical Unit class shall contain the following objects:

a)   at least one hierarchical logical unit or management logical unit;
b)   zero or more hierarchical logical units;
c)   zero or more management logical units;
d)   zero or more well known logical units;
e)   zero or more level 3 hierarchical logical units; and
f)   zero or more level 4 hierarchical logical units.

Each instance of a Level 3 Hierarchical Logical Unit class shall contain the following objects:

a)   at least one hierarchical logical unit or management logical unit;
b)   zero or more hierarchical logical units;
c)   zero or more management logical units;
d)   zero or more well known logical units; and
e)   zero or more level 4 hierarchical logical units.

Each instance of a Level 4 Hierarchical Logical Unit class shall contain the following objects:

a)   at least one hierarchical logical unit or management logical unit;
b)   zero or more hierarchical logical units;
c)   zero or more management logical units; and
d)   zero or more well known logical units.

Hierarchical logical units, management logical units, and well known logical units at each level in the hierarchical LUN structure are referenced by one of the following address methods:

a)   peripheral device address method (see 4.7.7.2);
b)   flat space addressing method (see 4.7.7.3);
c)   logical unit addressing method (see 4.7.7.4); or
d)   extended logical unit addressing method (see 4.7.7.5).

All LUNs, except LUN 0 (see 4.7.4), default to vendor specific values. All addressable entities, except well known logical units (see 4.6.17), may default to vendor specific values or may be defined by an application client (e.g., by the use of SCC-2 configuration commands (see SCC-2)).

Within the hierarchical LUN structure there may be SCSI devices each of which contain a SCSI target device that:

a)   has multiple logical units that are accessible through SCSI target ports in one SCSI domain; and
b)   transfers SCSI operations to a SCSI target device in another SCSI domain through a SCSI initiator device and its associated SCSI initiator ports.

When using the peripheral device addressing method or the logical unit addressing method the SCSI domains accessed by these SCSI initiator ports are referred to as buses. A SCSI target device that has SCSI devices attached to these buses shall assign numbers, other than zero, to those buses. The bus numbers shall be used as components of the LUNs to the logical units attached to those buses, as described in 4.7.7.2 and 4.7.7.4.

When using the peripheral device addressing method or the logical unit addressing method SCSI devices shall assign a bus number of zero to all the logical units within the SCSI target device that are not connected to another SCSI domain.

4.6.12 Level 2 Hierarchical Logical Unit class

The Level 2 Hierarchical Logical Unit class (see figure 18) contains:

   a)   the Hierarchical Logical Unit class;
   b)   the Management Logical Unit class;
   c)   the Well Known Logical Unit class; and
   d)   the Level 3 Hierarchical Logical Unit class (see 4.6.13).

Instantiation requirements for the Level 2 Hierarchical Logical Unit class are described in 4.6.11.

The Level 2 Hierarchical Logical Unit class is at level 2 within the hierarchical LUN structure.

All hierarchical logical units, management logical units, and well known logical units contained within a level 2 hierarchical logical unit shall have a Dependent Logical Unit attribute (see 4.6.18.4).

4.6.13 Level 3 Hierarchical Logical Unit class

The Level 3 Hierarchical Logical Unit class (see figure 18) contains:

   a)   the Hierarchical Logical Unit class;
   b)   the Management Logical Unit class;
   c)   the Well Known Logical Unit class; and
   d)   the Level 4 Hierarchical Logical Unit class (see 4.6.14).

Instantiation requirements for the Level 3 Hierarchical Logical Unit class are described in 4.6.11.

The Level 3 Hierarchical Logical Unit class is placed at level 3 within the hierarchical LUN structure.

All hierarchical logical units, management logical units, and well known logical units contained within a level 3 hierarchical logical unit shall have a Dependent Logical Unit attribute (see 4.6.18.4).

4.6.14 Level 4 Hierarchical Logical Unit class

The Level 4 Hierarchical Logical Unit class (see figure 18) contains:

   a)   the Hierarchical Logical Unit class;
   b)   the Management Logical Unit class; and
   c)   the Well Known Logical Unit class.

Instantiation requirements for the Level 4 Hierarchical Logical Unit class are described in 4.6.11.

The Level 4 Hierarchical Logical Unit class is placed at level 4 within the hierarchical LUN structure.

All hierarchical logical units, management logical units, and well known logical units contained within a level 4 hierarchical logical unit shall have a Dependent Logical Unit attribute (see 4.6.18.4).

4.6.15 Hierarchical Logical Unit class

The Hierarchical Logical Unit class (see figure 19) is a Logical Unit class (see 4.6.18) with the additional characteristics defined in this subclause.

The Hierarchical Logical Unit class (see figure 19) may be substituted with:

   a)   the Management Logical Unit class (see 4.6.16); or
   b)   the Well Known Logical Unit class (see 4.6.17).

A hierarchical logical unit is contained in a level 1 hierarchical logical unit (see 4.6.11), level 2 hierarchical logical unit (see 4.6.12), level 3 hierarchical logical unit (see 4.6.13), or level 4 hierarchical logical unit (see 4.6.14).

A hierarchical logical unit:

   a)   shall have one or more LUNs; and
   b)   may have a Dependent Logical Unit attribute (see 4.6.18.4).

4.6.16 Management Logical Unit class

The Management Logical Unit class (see figure 19) is a Hierarchical Logical Unit class (see 4.6.15) with the additional characteristics defined in this subclause.

A management logical unit:

    a)   shall support access to management functions (see FCP-4); and
    b)   may support other features only for the purposes of device management.

4.6.17 Well Known Logical Unit class

The Well Known Logical Unit class (see figure 19) is a Management Logical Unit class (see 4.6.15) with the additional characteristics defined in this subclause.

Well known logical units are addressed using the well known logical unit addressing method (see 4.7.7.5.1) of extended logical unit addressing (see 4.7.7.5). Each well known logical unit has one W-LUN. W-LUN values are defined in SPC-4.

If a SCSI target port receives a command or a task management function specifying a W-LUN and the well known logical unit specified by the W-LUN does not exist, the task router shall follow the rules for processing an incorrect logical unit number described in 5.11 and 7.12.

A well known logical unit shall support all the commands defined for it.

Access to well known logical units shall not be affected by access controls (see SPC-4).

All well known logical units:

    a)   shall not have logical unit names; and
    b)   shall identify themselves using the SCSI device names of the SCSI device in which they are contained.

      NOTE 1 - A SCSI target device may have multiple SCSI device names if the SCSI target device supports multiple SCSI transport protocols (see 4.6.9).

The name used to identify the well known logical unit is indicated in the Device Identification VPD pages designation descriptor for SCSI target devices (see SPC-4).

4.6.18 Logical Unit class

4.6.18.1 Logical Unit class overview

The Logical Unit class (see figure 19) contains:

    a)   the Device Server class (see 4.6.19);
    b)   the Task Manager class (see 4.6.21);
    c)   the Copy Manager class (see 4.6.20); and
    d)   the Task Set class (see 4.6.22).

The Logical Unit class (see figure 19) may be substituted with:

    a)   the Hierarchical Logical Unit class (see 4.6.15);
    b)   the Management Logical Unit class (see 4.6.16);
    c)   the Well Known Logical Unit class (see 4.6.17);
    d)   the Administrative Logical Unit class (see 4.6.25); or
    e)   the Subsidiary Logical Unit class (see 4.6.26).
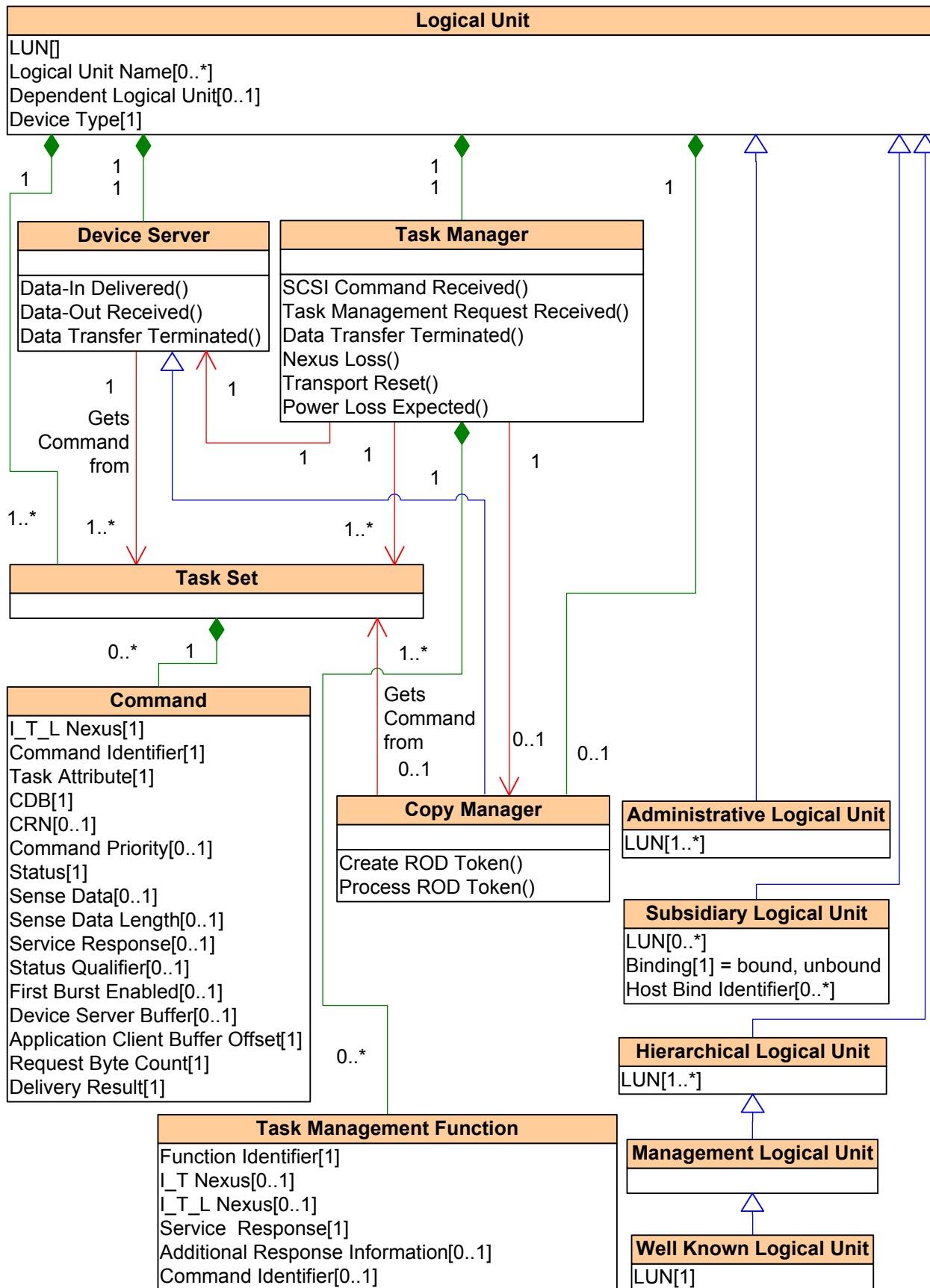
Figure 19 — Logical Unit class diagram

Each instance of a Logical Unit class shall contain the following objects:

    a) one device server;
    b) one task manager that shall contain:
        A) zero or more task management functions;
    c) zero or one copy manager; and
    d) one or more task sets each of which shall contain:
        A) zero or more commands.

The logical unit is the class to which commands are sent. One of the logical units within the SCSI target device shall be accessed using the LUN 0 or the REPORT LUNS well known logical unit.

If the portion of logical unit inventory that consists of administrative logical units and hierarchical logical units changes for any reason (e.g., completion of initialization, removal of a logical unit, or creation of a logical unit), then the device server shall establish a unit attention condition (see 5.14) for the SCSI initiator port associated with every I_T nexus, with the additional sense code set to REPORTED LUNS DATA HAS CHANGED unless otherwise specified by the applicable command standard.

If the portion of the logical unit inventory that consists of subsidiary logical units changes for any reason (e.g., removal of a subsidiary logical unit or creation of a subsidiary logical unit), then each subsidiary logical unit's device server in a logical unit conglomerate whose subsidiary logical unit inventory has changed shall establish a unit attention condition (see 5.14) for the SCSI initiator port associated with every I_T nexus, with the additional sense code set to REPORTED LUNS DATA HAS CHANGED unless otherwise specified by the applicable command standard. The unit attention conditions are coalesced as described in 5.14.2.

Data, if any, contained within a logical unit may be duplicated in whole or part on a different logical unit. The synchronization of that data between multiple logical units is outside the scope of this standard.

4.6.18.2 LUN attribute

The LUN attribute identifies the logical unit within a SCSI target device when accessed by a SCSI target port. If any logical unit within the scope of a SCSI target device includes one or more hierarchical logical units with a Dependent Logical Unit attribute (see 4.6.18.4) in its composition, then all LUNs within the scope of the SCSI target device, not contained in a logical unit conglomerate (see 4.6.10), shall have the format described in 4.7.6.3.

For any logical unit contained in a logical unit conglomerate, the LUNs, if any, associated with that logical unit shall have the format described in 4.7.6.2.

If there are no hierarchical logical units with a Dependent Logical Unit attribute and no logical unit conglomerates within the scope of the SCSI target device, then all LUNs should have the format described in 4.7.5.

The 64-bit or 16-bit quantity called a LUN is the value assigned to the LUN attribute defined by this standard. The fields containing the acronym LUN that compose the LUN attribute are historical nomenclature anomalies, not LUN attributes. LUN attributes having different values represent different LUNs, regardless of any implications to the contrary in 4.7 (e.g., LUN 00000000 00000000h is a different LUN from LUN 40000000 00000000h and LUN 00FF0000 00000000h is a different LUN from LUN 40FF0000 00000000h).

The multiplicity of the LUN attribute is specified by the Administrative Logical Unit class (see 4.6.11), Subsidiary Logical Unit class (see 4.6.26), Hierarchical Logical Unit class (see 4.6.15), and Well Known Logical Unit class (see 4.6.17).

See 4.7 for a definition of:

    a) the construction of LUNs to be used by SCSI target devices; and
    b) incorrect logical unit numbers.

4.6.18.3 Logical Unit Name attribute

The Logical Unit Name attribute identifies a name for a logical unit that is not a well known logical unit. A logical unit name shall be world wide unique. A logical unit name shall never change and may be used for persistent identification of a logical unit.

Logical unit name(s) are required as follows:

    a)　one or more logical unit names if the logical unit is not a well known logical unit; or
    b)　zero logical unit names if the logical unit is a well known logical unit.

The name used to identify the logical unit is reported in the logical unit name designation descriptor in the Device Identification VPD page (see SPC-4).

4.6.18.4 Dependent Logical Unit attribute

The Dependent Logical Unit attribute identifies a hierarchical logical unit that is:

    a)　addressed within a hierarchical LUN structure when that hierarchical logical unit resides at a lower numbered level in the hierarchy (i.e., when more than one level of hierarchy is used, no hierarchical logical unit within level 1 contains a Dependent Logical Unit attribute while all hierarchical logical units within level 2, level 3, and level 4 do contain a Dependent Logical Unit attribute); or
    b)　addressed with the logical unit addressing method (see 4.7.7.4) via another logical unit at the same numbered level in the hierarchy.

Any instance of a Hierarchical Logical Unit class that contains a Dependent Logical Unit attribute shall utilize the hierarchical LUN structure defined in 4.7.6.3. If any hierarchical logical unit within a SCSI target device includes a Dependent Logical Unit attribute, then:

    a)　all hierarchical logical units within the SCSI target device shall format all LUNs as described in 4.7.6; and
    b)　LUN 0, if any, or the REPORT LUNS well known logical unit, if any, (see SPC-4) shall set the HISUP bit to one in the standard INQUIRY data.

4.6.18.5 Device Type attribute

The Device Type attribute identifies the device type (e.g., direct access block device or sequential-access device) implemented by the logical unit as specified by the PERIPHERAL DEVICE TYPE field in the standard INQUIRY data (see SPC-5) and VPD parameters (see command standards and protocol standards).

4.6.19 Device Server class

4.6.19.1 Device Server class overview

The Device Server class (see figure 19) processes commands (see 4.6.23) from the task set (see 4.6.22).

The Device Server class may be substituted with the Copy Manager class (see 4.6.20).

The device server invokes:

    a)　the Send Data-In operation (see 4.6.6.4) of a SCSI target port to send data to the data-in buffer;
    b)　the Receive Data-Out operation (see 4.6.6.5) of a SCSI target port to receive data from the data-out buffer;
    c)　the Terminate Data Transfer operation (see 4.6.6.6) of a SCSI target port to terminate data transfers for a command that it is aborting; and
    d)　the Send Command Complete operation (see 4.6.6.7) of a SCSI target port to complete processing a command.

4.6.19.2 Data-In Delivered operation

The Data-In Delivered operation implements the Data-In Delivered SCSI transport protocol service confirmation (see 5.4.3.2.2) by completing processing of sent data.

4.6.19.3 Data-Out Received operation

The Data-Out Received operation implements the Data-Out Received SCSI transport protocol service confirmation (see 5.4.3.3.1) by completing processing of received data.

4.6.19.4 Data Transfer Terminated operation

The Data Transfer Terminated operation implements the Data Transfer Terminated SCSI transport protocol service confirmation (see 5.4.3.4.2) by completing processing of a data transfer termination.

4.6.20 Copy Manager class

4.6.20.1 Copy Manager class overview

The Copy Manager class (see figure 19) is a kind of application client and a kind of device server (see figure 12) that processes third-party copy commands (see SPC-4) from the task set (see 4.6.22) and their associated copy operations (see SPC-4).

4.6.20.2 Create ROD Token operation

The Create ROD Token operation creates a ROD token (see SPC-4) and returns it to an application client as a representation of specified data that may be exchanged with other application clients.

4.6.20.3 Process ROD Token operation

The Process ROD Token operation converts a ROD token (see SPC-4) received from an application client into the data represented by the ROD token.

4.6.21 Task Manager class

4.6.21.1 Task Manager class overview

The Task Manager class (see figure 19) contains:

    a)   the Task Management Function class (see 4.6.24).

Instantiation requirements for the Task Manager class are described in 4.6.18.1.

Instantiation requirements for the Task Manager class are also described in 4.6.18.1.

The Task Manager class (see figure 19) processes task management functions (see 4.6.24).

The task manager invokes:

    a)   the Terminate Data Transfer operation (see 4.6.6.6) of a SCSI target port to terminate data transfers for a command that the task manager is aborting;

    b)   the Task Management Function Executed operation (see 4.6.6.8) of a SCSI target port to complete processing a task management function;

    c)   the Terminate Data Transfer operation (see 4.6.6.6) of a SCSI target port to terminate data transfers for a command that it is aborting; and

    d)   the Send Command Complete operation (see 4.6.6.7) of a SCSI target port to complete processing a command.

4.6.21.2 SCSI Command Received operation

The SCSI Command Received operation implements the SCSI Command Received SCSI transport protocol service indication (see 5.4.2.3).

To process the SCSI Command Received operation, the task manager creates a command (see 4.6.23) based on the I_T_L Nexus argument, Command Identifier argument, CDB argument, Task Attribute argument, CRN argument, Command Priority argument, and First Burst Enabled argument and places the command into a task set (see 4.6.22).

4.6.21.3 Task Management Request Received operation

The Task Management Request Received operation implements the Task Management Request Received SCSI transport protocol service indication (see 7.12.3).

To process the Task Management Request Received operation, the task manager creates a task management function (see 4.6.24) based on the I_T_L Nexus argument, I_T Nexus argument, Command Identifier argument, and Function Identifier argument.

4.6.21.4 Data Transfer Terminated operation

The Data Transfer Terminated operation implements the Data Transfer Terminated SCSI transport protocol service confirmation (see 5.4.3.4.2) by completing processing of a data transfer termination.

4.6.21.5 Nexus Loss operation

The Nexus Loss operation implements the Nexus Loss SCSI transport protocol service indication (see 6.4.2) by processing an I_T nexus loss (see 6.3.4).

4.6.21.6 Transport Reset operation

The Transport Reset operation implements the Transport Reset SCSI transport protocol service indication (see 6.4.3) by processing a hard reset (see 6.3.2).

4.6.21.7 Power Loss Expected operation

The Power Loss Expected operation implements the Power Loss Expected SCSI transport protocol service indication (see 6.4.4) by processing a power loss expected (see 6.3.5).

4.6.22 Task Set class

The Task Set class (see figure 19) contains:

    a)   the Command class (see 4.6.23).

Instantiation requirements for the Task Set class are described in 4.6.18.1.

Instantiation requirements for the Task Set class are also described in 4.6.18.1.

The interactions among the commands in a task set are determined by the requirements for task set management specified in clause 8 and the ACA requirements specified in 5.9.   The number of task sets per logical unit and the boundaries between task sets are governed by the TST field in the Control mode page (see SPC-4).

4.6.23 Command class

4.6.23.1 Command class overview

The Command class (see figure 19) represents a request to a logical unit to do work.

The command persists until a device server, a task manager, or a task router invokes the Send Command Complete operation (see 4.6.6.7). For an example of the processing for a command see 5.7.

4.6.23.2 I_T_L Nexus attribute

The I_T_L Nexus attribute contains an identifier that represents the I_T_L nexus through which the command was received.

The Command class constructs the I_T_L Nexus attribute using the following attributes:

    a)   Initiator Port Identifier (see 4.6.8.2);
    b)   Target Port Identifier (see 4.6.6.2); and
    c)   LUN (see 4.6.18.2).

4.6.23.3 Command Identifier attribute

The Command Identifier attribute contains the identifier of the command (see 4.6.31.3).

4.6.23.4 Task Attribute attribute

A Task Attribute attribute (see 8.4) contains the task attribute (e.g., SIMPLE task attribute, ORDERED task attribute, HEAD OF QUEUE task attribute, ACA task attribute) of a command.

4.6.23.5 CDB attribute

The CDB attribute contains a CDB (see 5.2 and SPC-4) that defines the work to be performed by a logical unit.

4.6.23.6 CRN attribute

The CRN attribute, if any, contains the CRN of the command (see 5.1).

4.6.23.7 Command Priority attribute

The Command Priority attribute, if any, contains the priority of the command (see 8.5).

4.6.23.8 Status attribute

The Status attribute contains the status for the completed command (see 5.3).

4.6.23.9 Sense Data attribute

The Sense Data attribute, if any, contains the sense data for the completed command (see 5.13).

4.6.23.10 Sense Data Length attribute

The Sense Data Length attribute, if any, contains the length of the sense data for the completed command (see 5.13).

4.6.23.11 Service Response attribute

The Service Response attribute, if any, contains the service response for the completed command (see 5.1).

4.6.23.12 Status Qualifier attribute

The Status Qualifier attribute, if any, contains additional status information for the completed command (see 5.3.2 and 5.4.2.5).

4.6.23.13 First Burst Enabled attribute

The First Burst Enabled attribute, if any, specifies that first burst for the command is enabled (see 5.4.3).

4.6.23.14 Device Server Buffer attribute

The Device Server Buffer attribute, if any, contains the Device Server Buffer argument used for **Send Data-In** procedure calls (see 5.4.3.2.1) and **Receive Data-Out** procedure calls (see 5.4.3.3.1).

4.6.23.15 Application Client Buffer Offset attribute

The Application Client Buffer Offset attribute contains the Application Client Buffer Offset argument used for **Send Data-In** procedure calls (see 5.4.3.2.1) and **Receive Data-Out** procedure calls (see 5.4.3.3.1).

4.6.23.16 Request Byte Count attribute

The Request Byte Count attribute contains the Request Byte Count argument used for **Send Data-In** procedure calls (see 5.4.3.2.1) and **Receive Data-Out** procedure calls (see 5.4.3.3.1).

4.6.23.17 Delivery Result attribute

The Delivery Result attribute contains the Delivery Result argument from a **Data-In Delivered** procedure call (see 5.4.3.2.2) or a **Data-Out Received** procedure call (see 5.4.3.3.2).

4.6.24 Task Management Function class

4.6.24.1 Task Management Function class overview

The Task Management Function class (see figure 19) represents a SCSI task management function (see clause 7).

The task management function persists until a task manager invokes the Task Management Function Executed operation (see 4.6.6.8). For an example of the processing for a task management function see 7.13.

4.6.24.2 Function Identifier attribute

The Function Identifier attribute contains the function identifier (see 7.12).

4.6.24.3 I_T Nexus attribute

The I_T Nexus attribute contains an identifier that represents the I_T nexus affected by the task management function (see table 55).

The Task Management Function class constructs the I_T nexus attribute using the following attributes:

    a) Initiator Port Identifier (see 4.6.8.2); and
    b) Target Port Identifier (see 4.6.6.2).

4.6.24.4 I_T_L Nexus attribute

The I_T_L Nexus attribute contains an identifier that represents the I_T_L nexus, if any, affected by the task management function (see table 55).

The Task Management Function class constructs the I_T_L Nexus attribute using the following attributes:

    a) Initiator Port Identifier (see 4.6.8.2);
    b) Target Port Identifier (see 4.6.6.2); and
    c) LUN (see 4.6.18.2).

4.6.24.5 Command Identifier attribute

The Command Identifier attribute, if any, contains the command identifier of the command affected by the task management function (see table 55).

4.6.24.6 Service Response attribute

The Service Response attribute contains the service response for the completed task management function (see 7.12.4).

4.6.24.7 Additional Response Information attribute

The Additional Response Information attribute, if any, contains any additional response information for the task management function (see clause 7).

4.6.25 Administrative Logical Unit class

4.6.25.1 Administrative Logical Unit class overview

The Administrative Logical Unit class is a Logical Unit class (see 4.6.18) with the additional characteristics defined in 4.6.11.

The administrative functionality provided by an administrative logical unit shall not affect logical units outside the logical unit conglomerate.

An administrative logical unit shall have one or more LUNs.

An administrative logical unit shall not:

   a)   contain the Dependent Logical Unit attribute (see 4.6.18.4); or
   b)   be a well known logical unit (see 4.6.17).

4.6.25.2 Rerouted commands and task management functions

4.6.25.2.1 Overview

The task router (see 4.6.7) may route a command or task management function from subsidiary logical units to the administrative logical unit for the following reasons:

   a)   the command is associated with a subsidiary logical unit that is not valid (see 4.6.25.2.2);
   b)   the rerouting of a command was requested by the administrative logical unit (see 4.6.25.2.3); and
   c)   the rerouting of all task management functions was requested by the administrative logical unit (see 4.6.25.2.4).

The administrative logical unit's device server and task manager shall monitor the I_T_L Nexus argument and command identifier, if any, for each received command and task management function to detect the delivery of a command or task management function that is associated with a subsidiary logical unit.

4.6.25.2.2 Commands rerouted due to incorrect logical unit selection

If task router processing causes an administrative logical unit's device server to receive a command associated with an incorrect logical unit number for a subsidiary logical unit (see 4.6.7.2), then the device server:

   a)   shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and with the additional sense code set to SUBSIDIARY LOGICAL UNIT NOT CONFIGURED;
   b)   shall ignore the value of the NACA bit in the CDB; and
   c)   shall not establish an ACA condition in the administrative logical unit based on the delivery of a CHECK CONDITION related to an incorrect logical unit number for a subsidiary logical unit.

4.6.25.2.3 Command rerouting in response to an administrative logical unit request

An administrative logical unit may manage its logical unit conglomerate (e.g., for unit attention coalescing (see 5.14.2)) by requesting the task manager reroute to the administrative logical unit's device server one command that an application client sends to a device server in one of the subsidiary logical units. The administrative logical unit's device server causes one command to be rerouted by invoking the Reroute Conglomerate Command operation (see 4.6.7.4) in the task router (see 4.6.7).

If a command associated with a subsidiary logical unit is rerouted as a result of the processing of a Reroute Conglomerate Command operation, then the administrative logical unit's device server processes the command as follows:

  a) if:
      A) the command is allowed to terminate with CHECK CONDITION status (e.g., commands other than the INQUIRY command);
      B) the subsidiary logical unit with which the command is associated is not in a condition that requires the return of a status that has a higher status precedence (see 5.3.3) than that of the highest precedence unit attention (see 5.14.1) in the administrative device server's coalesced unit attentions queue (see 5.14.2); and
      C) the subsidiary logical unit with which the command is associated does not have a unit attention condition in its queue that has a higher unit attention precedence (see 5.14.1) than that of the highest precedence unit attention in the administrative device server's coalesced unit attentions queue,

      then the administrative logical unit's device server shall:

      A) not begin processing for the command;
      B) terminate the command with CHECK CONDITION status and set the sense data to indicate the appropriate logical unit conglomerate condition (e.g., a unit attention condition); and
      C) if the NACA bit is set to one in the CDB (see 5.2), then establish an ACA condition (see 5.9) for the subsidiary logical unit associated with the command;

      or

  b) if:
      A) the command is not allowed to terminate with CHECK CONDITION status (e.g., an INQUIRY command);
      B) the subsidiary logical unit with which the command is associated is in a condition that requires the return of a status that has a higher status precedence (see 5.3.3) than that of the highest precedence unit attention condition (see 5.14.1) in the administrative device server's coalesced unit attentions queue (see 5.14.2); or
      C) the subsidiary logical unit with which the command is associated has a unit attention condition in its queue that has a higher unit attention precedence (see 5.14.1) than that of the highest precedence unit attention condition in the administrative device server's coalesced unit attentions queue,

      then the administrative logical unit's device server:

      1) shall invoke the Route Command operation (see 4.6.7.2) to route the command to the correct subsidiary logical unit; and
      2) may invoke the Reroute Conglomerate Command operation (see 4.6.7.4) to request that the task router reroute another command associated with a subsidiary logical unit.

4.6.25.2.4 Task management function rerouting in response to an administrative logical unit request

An administrative logical unit may manage its logical unit conglomerate (e.g., for unit attention coalescing (see 5.14.2)) by requesting the task manager to reroute to the administrative logical unit's task manager zero or more task management function requests that an application client sends to a task manager in any of the subsidiary logical units. The administrative logical unit's device server causes zero or more task management function requests to be rerouted by invoking the following task router operations:

a) Reroute Conglomerate Task Management Functions operation (see 4.6.7.5) to begin the rerouting of task management function requests, if any; and
b) Stop Conglomerate Task Management Functions Rerouting operation (see 4.6.7.6) to stop the rerouting of task management function requests.

If a task management function request associated with a subsidiary logical unit is rerouted as a result of the processing of a Reroute Conglomerate Task Management Functions operation, then the administrative logical unit's task manager processes the task management function as follows:

a) if the task management function is a QUERY ASYNCHRONOUS EVENT, then the administrative logical unit's task manager shall process the QUERY ASYNCHRONOUS EVENT as described in 7.10; or
A) if the task management function is not a QUERY ASYNCHRONOUS EVENT (see 7.10), then the administrative logical unit's task manager shall cause the task management function to be processed by invoking the Task Management Request Received operation (see 4.6.21.3) in the task manager of the subsidiary logical unit associated with the task management function.

4.6.26 Subsidiary Logical Unit class

4.6.26.1 Subsidiary Logical Unit class overview

The Subsidiary Logical Unit class is a Logical Unit class (see 4.6.18) with the additional characteristics defined in 4.6.26.

A subsidiary logical unit:

a) may be contained in one or more logical unit conglomerates (see 4.6.10); and
b) has zero or more LUNs.

If a subsidiary logical unit has no LUNs:

a) the task sets are empty (i.e., contain no commands); and
b) the task manager contains no task management functions.

A subsidiary logical unit shall not:

a) contain the Dependent Logical Unit attribute (see 4.6.18.4); or
b) be a well known logical unit (see 4.6.17).

4.6.26.2 Binding attribute

The Binding attribute identifies a Subsidiary Logical Unit class as:

a) bound (i.e., has a relationship between a subsidiary logical unit, an administrative logical unit, and an application client identified by a host bind identifier as defined in SPC-5); or
b) unbound (i.e., has no assigned relationship between a subsidiary logical unit, an administrative logical unit, and an application client as defined in SPC-5).

4.6.26.3 Host Bind Identifier attribute

The Host Bind Identifier attribute identifies the relationships, if any, between a subsidiary logical unit, an administrative logical unit, and an application client as described in SPC-5.

4.6.27 SCSI Initiator Device class

A SCSI Initiator Device class (see figure 20) is a SCSI Device class that contains:

    a)   the Application Client class (see 4.6.28); and

    b)   the Discovery class (see 4.6.32).
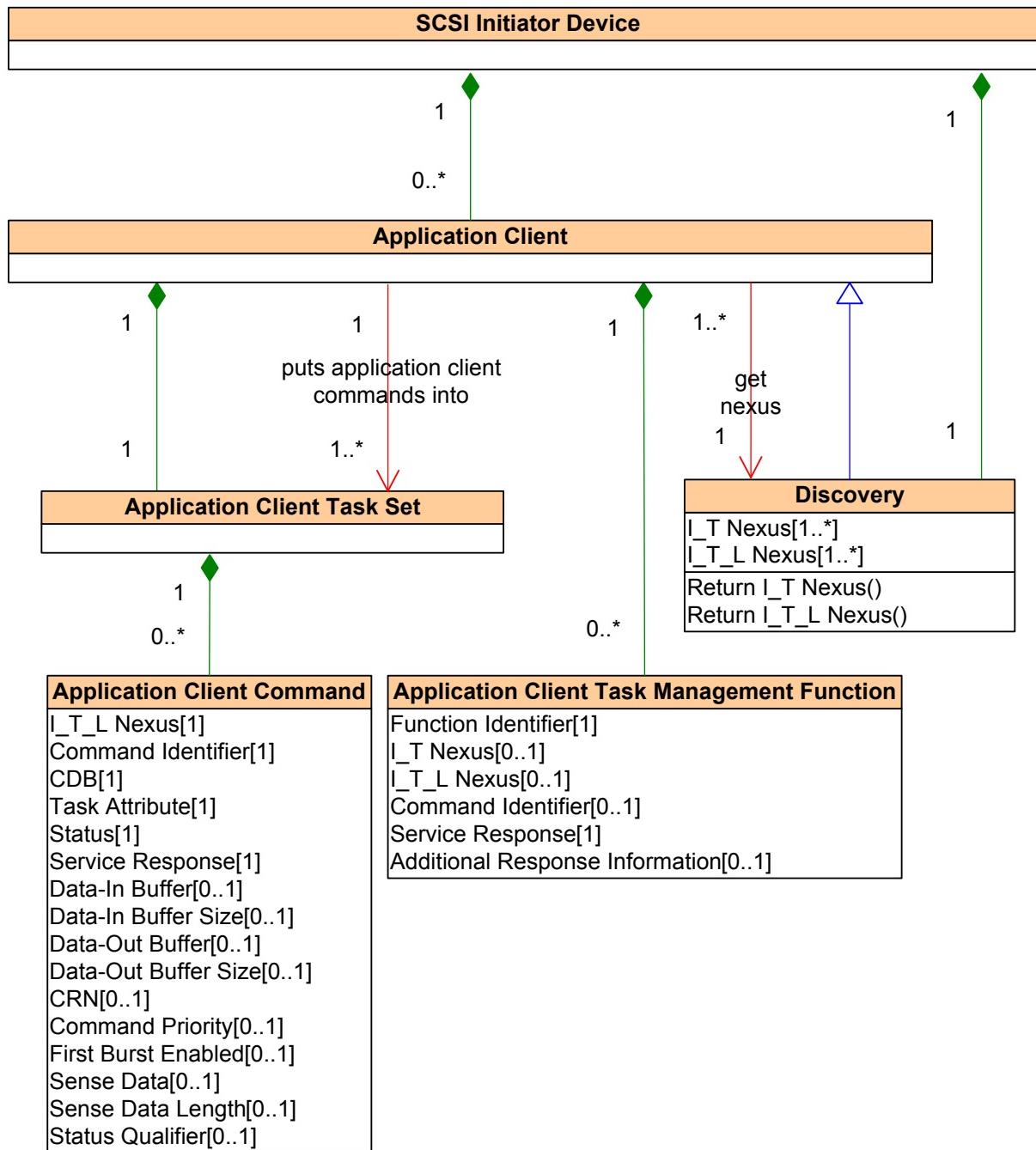
Figure 20 — SCSI Initiator Device class diagram

Each instance of a SCSI Initiator Device class shall contain the following objects:

    a)   one discovery; and

    b)   one or more application clients that contain:

    A) zero or more application client task management functions; and
    B) one application client task set that contains:
       a) zero or more application client commands.

The SCSI Initiator Device class is associated with the SCSI Initiator Port class (see figure 12).

4.6.28 Application Client class

4.6.28.1 Application Client class overview

An Application Client class (see figure 20) contains:

    a) the Application Client Task Management Function class (see 4.6.29); and
    b) the Application Client Task Set class (see 4.6.30).

The Application Client class may be substituted with the Discovery class (see 4.6.28).

Instantiation requirements for the Application Client class are described in 4.6.27.

Instantiation requirements for the Application Client Task Set class are described in 4.6.27.

An application client originates a command by:

    a) requesting I_T_L nexus information for that application client command by invoking the Return I_T_L
       Nexus operation (see 4.6.32.5) of the Discovery class;
    b) creating an application client command (see 4.6.31) and placing that application client command into
       the application client task set; and
    c) invoking the Send SCSI Command operation (see 4.6.8.4) of a SCSI initiator port.

The application client deletes an application client command from the application client task set after
determining that the command has completed according to the command lifetime rules in 5.5 (e.g., processing
the Command Complete Received operation).

An application client originates a task management function by:

    a) requesting I_T_L nexus information or I_T nexus information for that application client task
       management function by invoking the Return I_T Nexus operation (see 4.6.32.4) or the Return I_T_L
       Nexus operation (see 4.6.32.5) of the Discovery class;
    b) creating an application client task management function (see 4.6.29); and
    c) invoking the Send Task Management Request operation (see 4.6.8.5) of a SCSI initiator port.

The application client deletes an application client task management function after determining that the task
management function has completed according to the task management function lifetime rules in 7.11 (e.g.,
processing the Received Task Management Function Executed operation).

The application client may request processing of a task management function for:

    a) a logical unit through a request directed to the task manager within the logical unit; or
    b) all logical units known by a task router through a request directed to the task router within the SCSI
       target port.

The interactions between the task manager, task router, and the application client when a task management
request is processed are shown in 7.13.

4.6.28.2 Command Complete Received operation

The Command Complete Received operation implements the Command Complete Received SCSI transport
protocol service confirmation (see 5.4.2.5) by completing the processing of a command.

4.6.28.3 Received Task Management Function Executed operation

The Received Task Management Function Executed operation implements the Received Task Management
Function Executed SCSI transport protocol service confirmation (see 7.12.5) by completing the processing of
a task management function.

4.6.28.4 Nexus Loss operation

The Nexus Loss operation implements the Nexus Loss SCSI transport protocol service indication (see 6.4.2) by processing an I_T nexus loss (see 6.3.4).

4.6.28.5 Transport Reset operation

The Transport Reset operation implements the Transport Reset SCSI transport protocol service indication (see 6.4.3) by processing a hard reset (see 6.3.2).

4.6.29 Application Client Task Management Function class

4.6.29.1 Application Client Task Management Function class overview

The Application Client Task Management Function class (see figure 20) represents a SCSI task management function (see clause 7).

4.6.29.2 Function Identifier attribute

The Function Identifier attribute contains a function identifier (see 7.12).

4.6.29.3 I_T Nexus attribute

The I_T Nexus attribute contains an identifier that represents the I_T nexus affected by the task management function (see table 55).

The Application Client Task Management Function class constructs the I_T nexus attribute using the following attributes:

  a) Initiator Port Identifier (see 4.6.8.2); and
  b) Target Port Identifier (see 4.6.6.2).

4.6.29.4 I_T_L Nexus attribute

The I_T_L Nexus attribute contains an identifier that represents the I_T_L nexus, if any, affected by the task management function (see table 55).

The Application Client Task Management Function class constructs the I_T_L Nexus attribute using the following attributes:

  a) Initiator Port Identifier (see 4.6.8.2);
  b) Target Port Identifier (see 4.6.6.2); and
  c) LUN (see 4.6.18.2).

4.6.29.5 Command Identifier attribute

The Command Identifier attribute, if any, contains the command identifier of the command affected by the task management function (see table 55).

4.6.29.6 Service Response attribute

The Service Response attribute contains the service response (see clause 7).

4.6.29.7 Additional Response Information attribute

The Additional Response Information attribute, if any, contains any additional response information for the task management function (see clause 7).

4.6.30 Application Client Task Set class

The Application Client Task Set class (see figure 20) contains:

    a)   the Application Client Command class (see 4.6.31).

Instantiation requirements for the Application Client Task Set class are described in 4.6.27.

The interactions among the application client commands in an application client task set are not specified in this standard.

4.6.31 Application Client Command class

4.6.31.1 Application Client Command class overview

The Application Client Command class (see figure 20) represents a request to a logical unit to do work (see clause 5). A new command causes the creation of an application client command. The application client command persists until a command complete response is received or until the command is completed by a task management function or exception condition. For an example of the processing for a command see 5.7.

4.6.31.2 I_T_L Nexus attribute

The I_T_L Nexus attribute contains an identifier that represents the I_T_L nexus of the command (see 4.6.32.3).

The Application Client Command class constructs the I_T_L Nexus attribute using the following attributes:

    a)   Initiator Port Identifier (see 4.6.8.2);
    b)   Target Port Identifier (see 4.6.6.2); and
    c)   LUN (see 4.6.18.2).

4.6.31.3 Command Identifier attribute

A Command Identifier attribute is assigned by a SCSI initiator device to uniquely identify one command in the context of a particular I_T nexus or I_T_L nexus, allowing more than one command to be outstanding for that I_T_L nexus at the same time.

Each SCSI transport protocol defines the following limits on the command identifier to be used by SCSI ports that support that SCSI transport protocol:

    a)   the size of the command identifier, up to a maximum of 64 bits;
    b)   the scope of the command identifier (see table 13); and
    c)   whether the command identifier scope is shared with other uses (e.g., task management function identifiers).

Table 13 defines the command identifier scopes.

Table 13 — Command identifier scopes

Command identifier scope	Description	Application client rules [a]
I_T_L nexus	Each logical unit accessible through the SCSI target port has its own command identifiers.	The application client should have no more than one **Execute Command** procedure call in progress based on the command lifetime defined in 5.5 with: a) the same I_T_L nexus; and b) the same command identifier.
I_T nexus	All logical units accessible through the SCSI target port share command identifiers.	The application client should have no more than one **Execute Command** procedure call in progress based on the command lifetime defined in 5.5 with: a) the same I_T nexus; and b) the same command identifier.
[a] If the application client violates this recommendation, then the task manager or task router may detect an overlapped command (see 5.10).		

SCSI transport protocols may define additional restrictions on command identifier assignments (e.g., requiring command identifiers to be unique per I_T nexus or per I_T_L nexus, or sharing command identifier values with other uses such as task management functions).

4.6.31.4 CDB attribute

The CDB attribute contains a CDB (see 5.2 and SPC-4) that defines the work to be performed by a logical unit (see 5.4.2.2).

4.6.31.5 Task Attribute attribute

The Task Attribute attribute (see 8.4) contains the task attribute (e.g., SIMPLE task attribute, ORDERED task attribute, HEAD OF QUEUE task attribute, ACA task attribute) of a command (see 5.4.2.2).

4.6.31.6 Status attribute

The Status attribute contains the status (see 5.3) of the completed command (see 5.4.2.5).

4.6.31.7 Service Response attribute

The Service Response attribute contains the service response for the completed command (see 5.4.2.5).

4.6.31.8 Data-In Buffer attribute

The Data-In Buffer attribute, if any, contains the Data-In Buffer argument from an **Execute Command** procedure call (see 5.1).

4.6.31.9 Data-In Buffer Size attribute

The Data-In Buffer Size attribute, if any, contains the Data-In Buffer Size argument from an **Execute Command** procedure call (see 5.1).

4.6.31.10 Data-Out Buffer attribute

The Data-Out Buffer attribute, if any, contains the Data-Out Buffer argument from an **Execute Command** procedure call (see 5.1).

4.6.31.11 Data-Out Buffer Size attribute

The Data-Out Buffer Size attribute, if any, contains the Data-Out Buffer Size argument from an **Execute Command** procedure call (see 5.1).

4.6.31.12 CRN attribute

The CRN attribute, if any, contains the CRN of the command (see 5.1).

4.6.31.13 Command Priority attribute

The Command Priority attribute, if any, contains the priority (see 8.5) of the command (see 8.5).

4.6.31.14 First Burst Enabled attribute

The First Burst Enabled attribute, if any, specifies that first burst for the command is enabled (see 5.4.3.1).

4.6.31.15 Sense Data attribute

The Sense Data attribute, if any, contains the sense data (see 5.13) for the completed command (see 5.4.2.5).

4.6.31.16 Sense Data Length attribute

The Sense Data Length attribute, if any, contains the length of the sense data (see 5.13) for the completed command (see 5.4.2.5).

4.6.31.17 Status Qualifier attribute

The Status Qualifier attribute, if any, contains additional status information for the completed command (see 5.3.2).

4.6.32 Discovery class

4.6.32.1 Discovery class overview

The Discovery class (see figure 20) is an Application Client class (see 4.6.28) with the additional characteristics defined in this subclause.

The Discovery class queries:

    a) logical units using the REPORT LUNS command to determine LUNs;
    b) logical units and task routers using the INQUIRY command Device Identification VPD page (see SPC-4) to determine:
        A) logical unit names, if any;
        B) relative port identifiers, if any;
        C) target port identifiers; and
        D) target port names, if any;
    c) SCSI initiator ports to determine:
        A) initiator port identifiers by invoking the Get Initiator Port Identifier operation (see 4.6.8.6) of that SCSI initiator port; and
        B) initiator port names, if any, by invoking the Get Initiator Port Name operation (see 4.6.8.7) of that SCSI initiator port;

and

   d) the SCSI domain to determine:
       A) I_T nexuses within the SCSI domain; and
       B) I_T_L nexuses within the SCSI domain.

The method used by the Discovery class to determine the I_T nexus and I_T_L nexus within the SCSI domain may be defined by a SCSI transport protocol standard (e.g., discovery in SPL-3).

The Discovery class is associated with:

   a) the SCSI Initiator Port class (see figure 12); and
   b) the Logical Unit class (see figure 12).

4.6.32.2 I_T Nexus attribute

The I_T Nexus attribute represents a relationship between a SCSI initiator port, and a SCSI target port that defines task management function routing to be used between an application client and target port within a SCSI domain to transfer that task management function between a target port and an application client.

The Discovery class constructs the I_T nexus attribute using the following attributes:

   a) Initiator Port Identifier (see 4.6.8.2); and
   b) Target Port Identifier (see 4.6.6.2).

4.6.32.3 I_T_L Nexus attribute

The I_T_L Nexus attribute represents a relationship between a SCSI initiator port, a SCSI target port, and a logical unit that defines:

   a) command routing to be used between an application client and a device server within a SCSI domain to transfer that command and its associated data; and
   b) task management function routing between a logical unit and an application client to transfer that task management function.

The Discovery class constructs the I_T_L Nexus attribute using the following attributes:

   a) Initiator Port Identifier (see 4.6.8.2);
   b) Target Port Identifier (see 4.6.6.2); and
   c) LUN (see 4.6.18.2).

4.6.32.4 Return I_T Nexus operation

The Return I_T Nexus operation returns the requested I_T nexus attribute to the requesting application client.

4.6.32.5 Return I_T_L Nexus operation

The Return I_T_L Nexus operation returns the requested I_T_L Nexus attribute to the requesting application client.

4.7 Logical unit number (LUN)

4.7.1 Introduction

Subclause 4.7 defines the construction of LUNs to be used by SCSI target devices. Application clients should use only those LUNs returned by a REPORT LUNS command (see SPC-4). The task router shall respond to incorrect logical unit numbers (i.e., LUNs other than those reported by a REPORT LUNS command with the SELECT REPORTS field set to 02h) as described in 5.11 and 4.6.7.2.

4.7.2 Logical unit representation format

If an application client displays or otherwise makes a 64-bit LUN value visible, then the application client should display it in hexadecimal format with byte 0 first (i.e., on the left) and byte 7 last (i.e., on the right), regardless of the internal representation of the LUN value (e.g., a single level LUN with an ADDRESS METHOD field set to 01b (i.e., flat space addressing) and a FLAT SPACE LUN field set to 0001h should be displayed as 40 01 00 00 00 00 00 00h, not 00 00 00 00 00 00 01 40h). A separator (e.g., space, dash, or colon) may be included between each byte, each two bytes (e.g., 4001-0000-0000-0000h), or each four bytes (e.g., 40010000 00000000h).

If displaying a single level LUN structure using the peripheral device addressing method (see table 14) or a single level LUN structure using the flat space addressing method (see table 15), then an application client may display the value as a single 2-byte value representing only the first level LUN (e.g., 40 01h). A separator (e.g., space, dash, or colon) may be included between each byte.

If displaying a single level LUN structure using the extended flat space addressing method (see table 16), then an application client may display the value as a single 4-byte value representing only the first level LUN (e.g., D2 00 00 01h). A separator (e.g., space, dash, or colon) may be included between each byte, or between each two bytes (e.g., D200 0001h).

If displaying a single level LUN structure using the long extended flat space addressing method (see table 17), then an application client may display the value as a single 6-byte value representing only the first level LUN (e.g., E2 00 00 01 00 01h). A separator (e.g., space, dash, or colon) may be included between each byte, or between each two bytes (e.g., E200 0001 0001h).

If displaying a 16-bit LUN value, then an application client should display the value as a single 2-byte value (e.g., 40 01h). A separator (e.g., space, dash, or colon) may be included between each byte.

4.7.3 LUNs overview

All LUN formats described in this standard are complex in structure (see 4.7.6) or a single level structure (see 4.7.5). The HISUP bit shall be set to one in the standard INQUIRY data (see SPC-4) when any LUN format described in this standard is used.

The LUN identifier defined by a transport protocol standard shall contain 64 bits or 16 bits. For transport protocol standards that define 16-bit LUN identifiers:

   a) the two bytes shall use an addressing method (see 4.7.7) with a length of two bytes (e.g., see table 27, table 28, table 29, or table 32); and
   b) if the peripheral device addressing method (see 4.7.7.2) is used, then the BUS IDENTIFIER field shall be set to 00h.

All LUN identifiers defined by command standards shall contain 64 bits. If a 16-bit transport protocol LUN identifier is contained in a 64-bit LUN identifier defined by a command standard, then:

   a) the 16 bits of the LUN identifier defined by the transport protocol standard shall be placed in the high order 16 bits of the 64-bit LUN identifier; and
   b) all other bits of the 64-bit LUN identifier shall be set to zero.

4.7.4 Minimum LUN addressing requirements

All SCSI target devices shall support LUN 0 (i.e., 00000000 00000000h) or the REPORT LUNS well known logical unit. For SCSI target devices that support complex LUN structures (see 4.7.6) the LUN 0 or the REPORT LUNS well known logical unit shall be the logical unit that an application client addresses to determine information about the SCSI target device and the logical units contained within the SCSI target device.

If the SCSI target device supports logical unit conglomerates (see 4.6.10), then the application client may address the administrative logical unit (see 4.6.11) of a logical unit conglomerate to determine information about subsidiary logical units (see 4.6.26) in that logical unit conglomerate.

The responses to commands sent to incorrect logical units are defined in 5.11. The response to task management functions sent to incorrect logical units is defined in 7.1.

4.7.5 Single level LUN structure

Table 14 describes a single level subset of the format described in 4.7.6.3 for SCSI target devices that contain 256 or fewer logical units.

Table 14 — Single level LUN structure using peripheral device addressing method

Bit Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (00b)		BUS IDENTIFIER (00h)					
1	TARGET OR LUN							
2	Null second level LUN (0000h)							
3								
4	Null third level LUN (0000h)							
5								
6	Null fourth level LUN (0000h)							
7								

Byte 2 through byte 7 in an 8-byte single level LUN structure using the peripheral device addressing method shall each contain 00h (see table 14). The value in the TARGET OR LUN field shall address a single level logical unit and be between 0 and 255, inclusive. A value of 00b in the ADDRESS METHOD field specifies peripheral device addressing (see 4.7.7.2). A value of 00h in the BUS IDENTIFIER field specifies the current level (see 4.7.7.2).

Table 15 describes a single level subset of the format described in 4.7.6.3 for SCSI target devices that contain 16 384 or fewer logical units.

Table 15 — Single level LUN structure using flat space addressing method

Bit Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (01b)		(MSB)					
1	FLAT SPACE LUN							(LSB)
2	Null second level LUN (0000h)							
3								
4	Null third level LUN (0000h)							
5								
6	Null fourth level LUN (0000h)							
7								

Byte 2 through byte 7 in an 8-byte single level LUN structure using the flat space addressing method shall each contain 00h (see table 15). The value in the FLAT SPACE LUN field shall be between 0 and 16 383, inclusive. A value of 01b in the ADDRESS METHOD field specifies flat space addressing (see 4.7.7.3) at the current level.

Table 16 describes a single level subset of the format described in 4.7.6.3 for SCSI target devices that contain more than 16 384 logical units and less than 16 777 216 logical units.

Table 16 — Single level LUN structure using extended flat space addressing method

Bit Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (11b)		LENGTH (01b)		EXTENDED ADDRESS METHOD (2h)			
1	(MSB)							
•••	EXTENDED FLAT SPACE LUN							
3								(LSB)
4	Null second level LUN (0000h)							
5								
6	Null third level LUN (0000h)							
7								

Byte 4 through byte 7 in an 8-byte single level LUN structure using the extended flat space addressing method shall each contain 00h (see table 16). The value in the EXTENDED FLAT SPACE LUN field shall be between 0 and 16 777 215, inclusive. A value of 11b in the ADDRESS METHOD field with a value of 2h in the EXTENDED ADDRESS METHOD field specifies extended flat space addressing (see 4.7.7.5.2) at the current level. A value of 01b in the LENGTH field specifies that the LUN specified in the EXTENDED FLAT SPACE LUN field is three bytes in length.

Table 17 describes a single level subset of the format described in 4.7.6.3 for SCSI target devices that contain more than 16 777 216 logical units and less than 1 099 511 627 776 logical units.

Table 17 — Single level LUN structure using long extended flat space addressing method

Bit Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (11b)		LENGTH (10b)		EXTENDED ADDRESS METHOD (2h)			
1	(MSB)							
•••				LONG EXTENDED FLAT SPACE LUN				
5								(LSB)
6				Null second level LUN (0000h)				
7								

Byte 6 and byte 7 in an 8-byte single level LUN structure using the long extended flat space addressing method shall each contain 00h (see table 16). The value in the LONG EXTENDED FLAT SPACE LUN field shall be between 0 and 1 099 511 627 775, inclusive. A value of 11b in the ADDRESS METHOD field with a value of 2h in the EXTENDED ADDRESS METHOD field specifies extended flat space addressing (see 4.7.7.5.2) at the current level. A value of 10b in the LENGTH field specifies that the LUN specified in the LONG EXTENDED FLAT SPACE LUN field is five bytes in length.

The presence of well known logical units shall not affect the requirements defined within this subclause.

If a SCSI target device contains 256 or fewer logical units, none of which are dependent logical units (see 4.6.18.4), then the SCSI target device's LUNs:

    a)   should have the format shown in table 14 (i.e., peripheral device addressing);
    b)   may have the format shown in table 15 (i.e., flat space addressing);
    c)   may have the format shown in table 16 (i.e., extended flat space addressing); or
    d)   may have the format shown in table 17 (i.e., long extended flat space addressing).

If a SCSI target device contains more than 256 logical units and 16 384 or fewer logical units, none of which are dependent logical units (see 4.6.18.4), then the SCSI target device's LUNs:

    a)   should have the format shown in table 15 (i.e., flat space addressing);
    b)   may have the format shown in table 16 (i.e., extended flat space addressing);
    c)   may have the format shown in table 17 (i.e., long extended flat space addressing); or
    d)   may have the format shown in table 14 (i.e., peripheral device addressing) for up to 256 of the logical units within the SCSI target device.

If a SCSI target device contains more than 16 384 logical units and less than 16 777 216 logical units, none of which are dependent logical units (see 4.6.18.4), then the SCSI target device's LUNs:

    a)   should have the format shown in table 16 (i.e., extended flat space addressing);
    b)   may have the format shown in table 17 (i.e., long extended flat space addressing);
    c)   may have the format shown in table 15 (i.e., flat space addressing) for up to 16 384 of the logical units within the SCSI target device; or

    d)   may have the format shown in table 14 (i.e., peripheral device addressing) for up to 256 of the logical units within the SCSI target device.

If a SCSI target device contains more than 16 777 216 logical units and less than 1 099 511 627 776 logical units, none of which are dependent logical units (see 4.6.18.4), then the SCSI target device's LUNs:

    a)   should have the format shown in table 17 (i.e., long extended flat space addressing);
    b)   may have the format shown in table 16 (i.e., extended flat space addressing) for up to 16 777 216 of the logical units within the SCSI target device;
    c)   may have the format shown in table 15 (i.e., flat space addressing) for up to 16 384 of the logical units within the SCSI target device; or
    d)   may have the format shown in table 14 (i.e., peripheral device addressing) for up to 256 of the logical units within the SCSI target device.

4.7.6 Complex LUN structures

4.7.6.1 Complex LUN structures overview

The eight byte LUN structures that represent complex configurations of logical units (e.g., logical units in a multilevel hierarchy) are summarized in table 18.

Table 18 — Complex LUN structures

LUN structure	Associated class	Reference
Conglomerate	Logical Unit Conglomerate class (see 4.6.10)	4.7.6.2
Hierarchical	Level 1 Hierarchical Logical Unit class (see 4.6.11)	4.7.6.3

4.7.6.2 Logical unit conglomerate LUN structure

The logical unit conglomerate LUN structure is shown in table 19. Each logical unit conglomerate LUN structure shall contain eight bytes.

Table 19 — Logical unit conglomerate LUN structure

Bit Byte	7	6	5	4	3	2	1	0
0 ••• i	ADMINISTRATIVE ELEMENT (see table 20)							
i+1 ••• k	SUBSIDIARY ELEMENT (see table 20)							
k+1 ••• 7	PAD (if needed)							

The ADMINISTRATIVE ELEMENT field contains addressing information for an administrative logical unit (see 4.6.11). The ADMINISTRATIVE ELEMENT field has a length of two, four, or six bytes. The format of the ADMINISTRATIVE ELEMENT field is shown in table 20.

The SUBSIDIARY ELEMENT field contains:

   a) zero, when the logical unit conglomerate LUN structure identifies the administrative logical unit specified by the ADMINISTRATIVE ELEMENT field; or
   b) non-zero logical unit addressing information for a subsidiary logical unit (see 4.6.26) in the same logical unit conglomerate as the administrative logical unit specified by the ADMINISTRATIVE ELEMENT field.

The SUBSIDIARY ELEMENT field has a length of two, four, or six bytes. The format of the SUBSIDIARY ELEMENT field is shown in table 20.

If a SUBSIDIARY ELEMENT field does not contain logical unit addressing information for a subsidiary logical unit, then that field shall be set to zero.

The PAD field shall contain zero to four bytes set to zero such that the total length of the logical unit conglomerate LUN structure is eight bytes.

The format of addressing fields in the logical unit conglomerate LUN structure is shown in table 20.

Table 20 — Format of addressing fields in the logical unit conglomerate LUN structure

Bit Byte	7	6	5	4	3	2	1	0
k	ADDRESS METHOD							
k+m	ADDRESS METHOD SPECIFIC							

The ADDRESS METHOD field defines the contents of the ADDRESS METHOD SPECIFIC field. See table 21 for the address methods defined for the ADDRESS METHOD field in the logical unit conglomerate LUN structure.

Table 21 — ADDRESS METHOD field in the logical unit conglomerate LUN structure

Code	Description	Reference
00b	Simple logical unit addressing method	4.7.7.1
01b	Reserved	
10b	Reserved	
11b	Extended logical unit addressing method	4.7.7.5

4.7.6.3 Hierarchical LUN structure

The hierarchical LUN structure (see table 23) contains four levels of addressing fields. Each level shall use byte 0 and byte 1 to define the address and location of the SCSI target device to be addressed on that level.

If the LUN specifies that the command or task management function is to be relayed to the next level (i.e., peripheral device addressing method (see 4.7.7.2) is selected in byte 0 and byte 1 of the hierarchical LUN structure and the BUS IDENTIFIER field is set to a value greater than zero), then the current level shall use byte 0 and byte 1 of the hierarchical LUN structure to determine the address of the SCSI target device to which the command or task management function is to be sent. When the command or task management function is sent to the SCSI target device the hierarchical LUN structure that was received shall be adjusted to create a new hierarchical LUN structure (see table 22 and figure 21).

SCSI target devices shall keep track of the addressing information necessary to transmit information back through all intervening levels to the command's or task management function's originating SCSI initiator port.
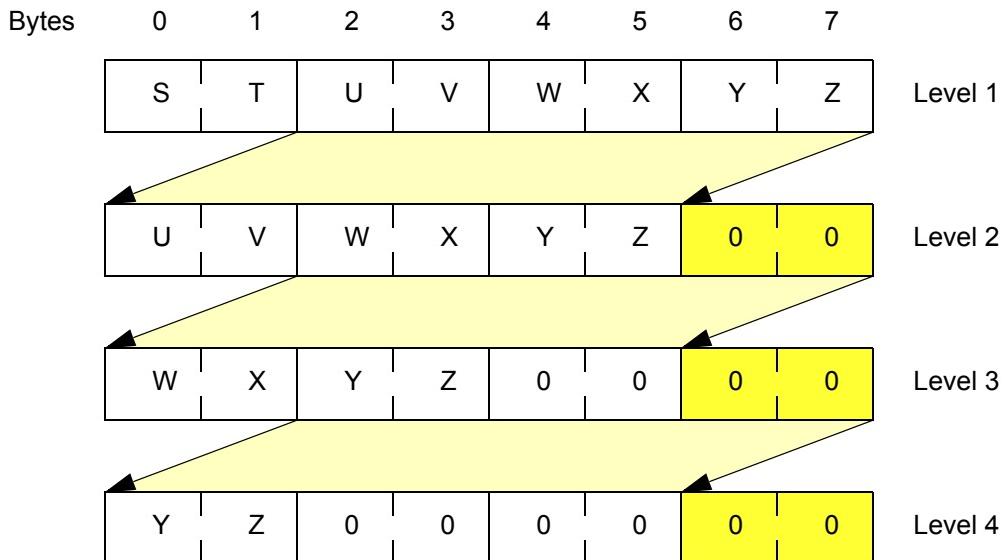
A LUN may use fewer than four levels of addressing fields. If fewer than four levels of addressing fields are used, then the size of the highest numbered level addressing field may be greater than two bytes (e.g., if a LUN uses three levels of addressing fields, the level three addressing field may consist of four bytes that contain WXYZ).

Figure 21 — Hierarchical LUN structure adjustments

Table 22 — Hierarchical LUN structure adjustments

Byte position		
Old		**New**
0 & 1	Moves to	Not Used
2 & 3	Moves to	0 & 1
4 & 5	Moves to	2 & 3
6 & 7	Moves to	4 & 5
n/a	zero fill	6 & 7

The hierarchical LUN structure requirements as viewed from the application client are shown in table 23.

Table 23 — Hierarchical LUN structure

Bit Byte	7	6	5	4	3	2	1	0
0				FIRST LEVEL ADDRESSING (see table 24)				
1								
2				SECOND LEVEL ADDRESSING (see table 24)				
3								
4				THIRD LEVEL ADDRESSING (see table 24)				
5								
6				FOURTH LEVEL ADDRESSING (see table 24)				
7								

The FIRST LEVEL ADDRESSING field specifies the first level address of a SCSI target device. See table 24 for a definition of the FIRST LEVEL ADDRESSING field.

The SECOND LEVEL ADDRESSING field specifies the second level address of a SCSI target device. See table 24 for a definition of the SECOND LEVEL ADDRESSING field.

The THIRD LEVEL ADDRESSING field specifies the third level address of a SCSI target device. See table 24 for a definition of the THIRD LEVEL ADDRESSING field.

The FOURTH LEVEL ADDRESSING field specifies the fourth level address of a SCSI target device. See table 24 for a definition of the FOURTH LEVEL ADDRESSING field.

Table 24 — Format of addressing fields in the hierarchical LUN structure

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD							
n+1	ADDRESS METHOD SPECIFIC							

The ADDRESS METHOD field defines the contents of the ADDRESS METHOD SPECIFIC field. See table 25 for the address methods defined for the ADDRESS METHOD field in the hierarchical LUN structure. The ADDRESS METHOD field only defines address methods for entities that are directly addressable by an application client.

Table 25 — ADDRESS METHOD field in the hierarchical LUN structure

Code	Description	Reference
00b	Peripheral device addressing method	4.7.7.2
01b	Flat space addressing method	4.7.7.3
10b	Logical unit addressing method	4.7.7.4
11b	Extended logical unit addressing method [a]	4.7.7.5
[a] Extended logical unit addresses have sizes of two bytes, four bytes, six bytes, or eight bytes. Extended logical unit addresses that are larger than two bytes shall be used only at the highest numbered level in the eight byte LUN structure (see figure 21) that is used by a LUN. Use of extended logical unit addresses shall not cause the total size of a LUN to exceed eight bytes.		

4.7.7 Addressing methods

4.7.7.1 Simple logical unit addressing method

The simple logical unit addressing method (see table 26) specifies a LUN element in a logical unit conglomerate LUN structure (see 4.7.6.2).

Table 26 — Simple logical unit addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (00b)		(MSB)					
n+1				SIMPLE LUN				(LSB)

The SIMPLE LUN field specifies a LUN element (i.e., administrative or subsidiary) in a logical unit conglomerate LUN structure.

4.7.7.2 Peripheral device addressing method

If the peripheral device addressing method (see table 27) is selected, the SCSI target device should relay the received command or task management function to the addressed dependent logical unit.

If the SCSI device does not relay any commands or task management functions to the addressed dependent logical unit, then the SCSI device shall follow the rules for processing an incorrect logical unit number described in 5.11 and 7.12.

If the SCSI device does relay some commands and task management functions to the addressed dependent logical unit, then the SCSI device shall:

  a) complete any command that is not relayed with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE; and

b)   terminate a task management function that is not relayed with a service response of SERVICE DELIVERY
     OR TARGET FAILURE.

NOTE 2 - A SCSI device may filter (i.e., not relay) commands or task management functions to prevent
operations with deleterious effects from reaching a dependent logical unit (e.g., a WRITE command directed
to a logical unit that is participating in a RAID volume).

Table 27 — Peripheral device addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (00b)		BUS IDENTIFIER					
n+1	TARGET OR LUN							

The BUS IDENTIFIER field identifies the bus or path that the SCSI device shall use to relay the received
command or task management function. The BUS IDENTIFIER field may use the same value encoding as the
BUS NUMBER field (see 4.7.7.4) with the most significant bits set to zero. However, if the BUS IDENTIFIER field is
set to 00h, then the command or task management function is to be relayed to a logical unit within the SCSI
target device at the current level.

The TARGET OR LUN field specifies the SCSI domain to which the SCSI device shall relay the received
command or task management function. The meaning and usage of the TARGET OR LUN field depends on
whether the BUS IDENTIFIER field contains zero.

A BUS IDENTIFIER field of zero specifies a logical unit at the current level. This representation of a logical unit
may be used either when the SCSI target device at the current level does not use hierarchical LUN structure
addressing for assigning LUNs to entities or when the SCSI target device at the current level includes entities
that are assigned LUNs but are not attached to SCSI buses. If the BUS IDENTIFIER field contains zero, then the
command or task management function shall be relayed to the current level logical unit specified by the
TARGET OR LUN field within or joined to the current level SCSI device.

A BUS IDENTIFIER field greater than zero represents a SCSI domain that connects a group of SCSI target
devices to the current level SCSI device. Each SCSI domain shall be assigned a unique bus identifier number
from 1 to 63. These bus identifiers shall be used in the BUS IDENTIFIER field when assigning addresses to
logical units contained within the attached SCSI domains. If the BUS IDENTIFIER field is greater than zero, then
the command or task management function shall be relayed to the logical unit within the SCSI target device
specified in the TARGET OR LUN field located in the SCSI domain specified by the BUS IDENTIFIER field with the
LUN being set to the contents of the received LUN shifted by two bytes as described in 4.7.6.3. The SCSI
target device information in the TARGET OR LUN field is a mapped representation of a target port identifier.

The SCSI target device located within the current level is addressed when the BUS IDENTIFIER field is set to
zero and the TARGET OR LUN field is set to zero, also known as LUN 0 (see 4.7.4).

Figure 22 shows the selection of a logical unit using the peripheral device addressing method.
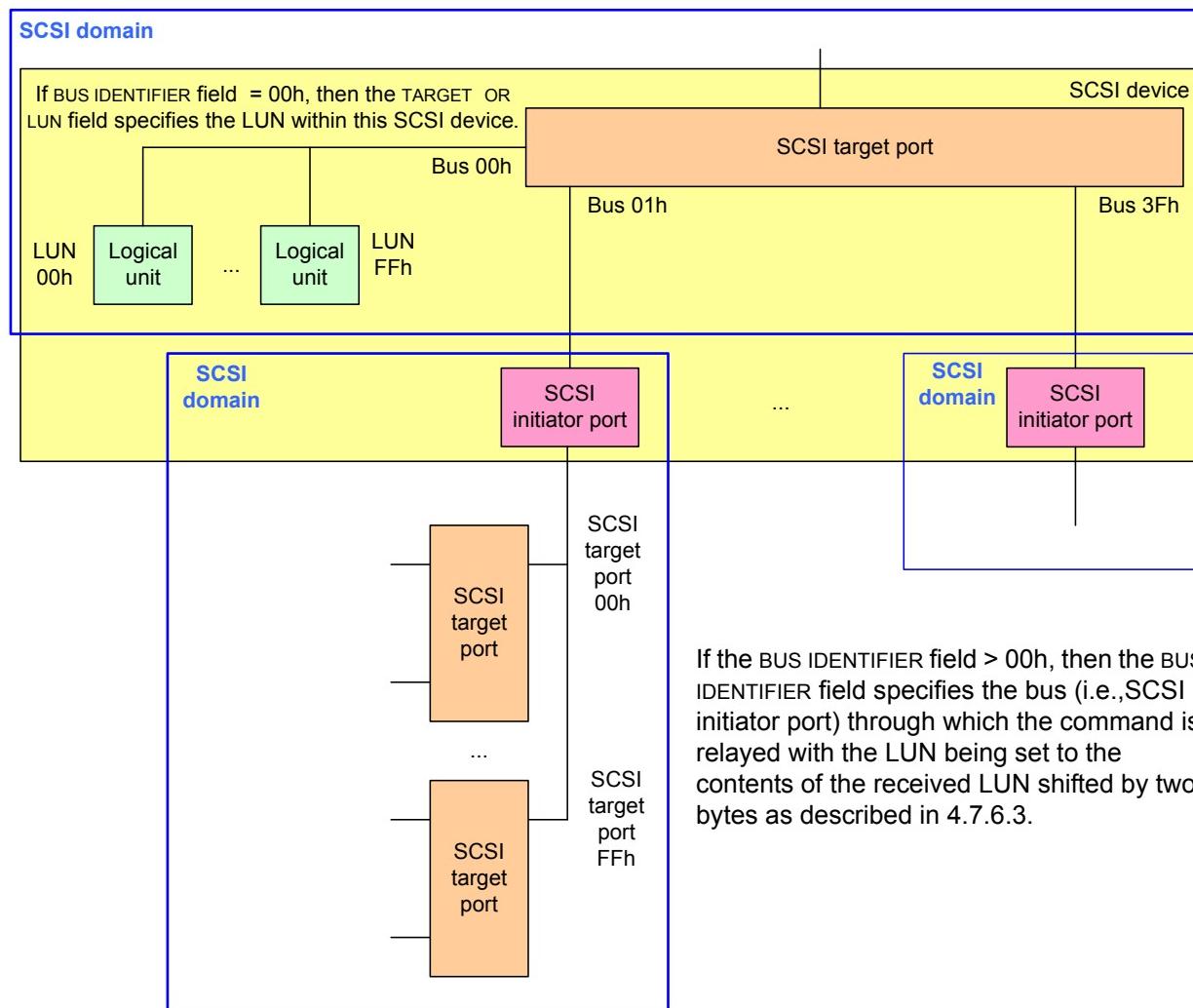
Figure 22 — Logical unit selection using the peripheral device addressing format

4.7.7.3 Flat space addressing method

The flat space addressing method (see table 28) specifies a logical unit at the current level.

The contents of all hierarchical LUN structure addressing fields following a flat space addressing method addressing field shall be ignored.

Table 28 — Flat space addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (01b)		(MSB)					
n+1			FLAT SPACE LUN					(LSB)

The FLAT SPACE LUN field specifies the current level logical unit.

4.7.7.4 Logical unit addressing method

If the logical unit addressing method (see table 29) is selected, the SCSI target device should relay the received command or task management function to the addressed dependent logical unit.

If the SCSI device does not relay any commands or task management functions to the addressed dependent logical unit, then the SCSI device shall follow the rules for processing an incorrect logical unit number described in 5.11 and 7.12.

If the SCSI device does relay some commands and task management functions to the addressed dependent logical unit, then the SCSI device shall:

    a) complete any command that is not relayed with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE; and

    b) terminate a task management function that is not relayed with a service response of SERVICE DELIVERY OR TARGET FAILURE.

> NOTE 3 - A SCSI device may filter (i.e., not relay) commands or task management functions to prevent operations with deleterious effects from reaching a dependent logical unit (e.g., a WRITE command directed to a logical unit that is participating in a RAID volume).

The contents of all hierarchical LUN structure addressing fields following a logical unit addressing method addressing field shall be ignored.

Table 29 — Logical unit addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (10b)		TARGET					
n+1	BUS NUMBER			LUN				

The TARGET field, BUS NUMBER field, and LUN field address the logical unit to which the received command or task management function shall be relayed. The command or task management function shall be relayed to the logical unit specified by the LUN field within the SCSI target device specified by the TARGET field located on the bus specified by the BUS NUMBER field. The value in the LUN field shall be placed in the least significant bits of the TARGET OR LUN field in a single level LUN structure for LUNs 255 and below (see 4.7.5). The TARGET field contains a mapped representation of a target port identifier.

Figure 23 shows the selection of a logical unit using the logical unit addressing method.
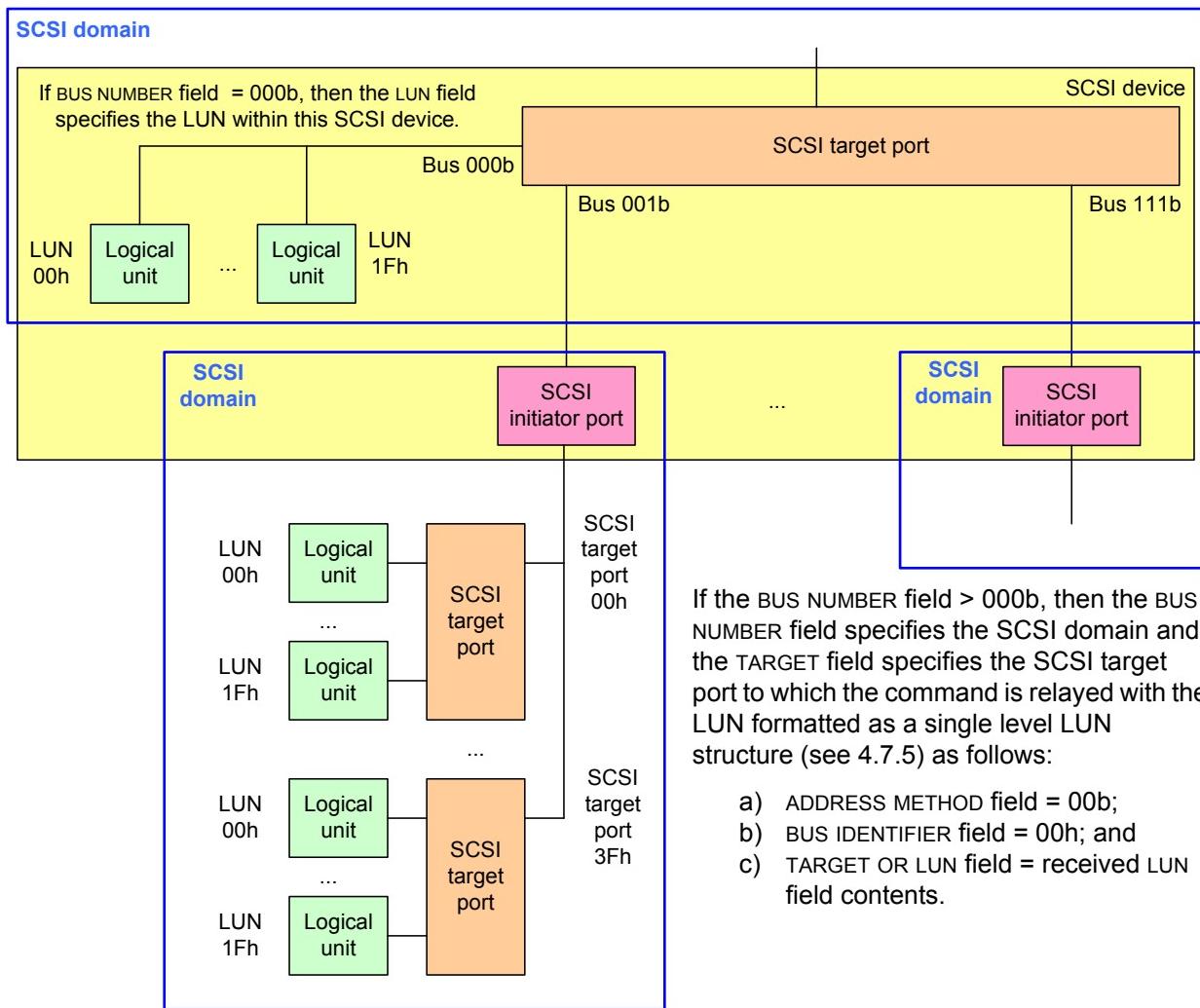
Figure 23 — Logical unit selection using the logical unit addressing format

4.7.7.5 Extended logical unit addressing

Extended logical unit addressing (see table 30) specifies a logical unit at the current level.

Extended logical unit addressing builds on the formats defined for dependent logical units (see 4.6.18.4) but may be used by SCSI devices having a single level logical unit structure or a multiple level logical unit structure. If addressing dependent logical units, then the logical unit information at each level, except the highest numbered level used, fits in exactly two bytes. Extended logical unit addresses have sizes of two bytes, four bytes, six bytes, or eight bytes. Use of extended logical unit addresses shall not cause the total size of a LUN to exceed eight bytes.

The contents of all hierarchical LUN structure addressing fields following an extended logical unit addressing method addressing field shall be ignored.

Extended logical units are identified by the ADDRESS METHOD field (see table 25 in 4.7.6.3) in the same manner as is the case for dependent logical units. An ADDRESS METHOD field value of 11b specifies the extended logical unit addressing method.

Table 30 — Extended logical unit addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH		EXTENDED ADDRESS METHOD			
n+1	(MSB)							
•••			EXTENDED ADDRESS METHOD SPECIFIC					
m								(LSB)

The LENGTH field (see table 31) specifies the length of the EXTENDED ADDRESS METHOD SPECIFIC field. A LUN that includes a LENGTH field value that goes beyond the LUN field length supported by the transport protocol is invalid and shall follow the rules for processing an incorrect logical unit number described in 5.11 and 7.12.

Table 31 — LENGTH field and related sizes

Code	Size in bytes of		Reference
	EXTENDED ADDRESS METHOD SPECIFIC field	Extended logical unit addressing format	
00b	1	2	table 32
01b	3	4	table 33
10b	5	6	table 34
11b	7	8	table 35

Table 32, table 33, table 34, and table 35 show the four extended logical unit addressing formats.

Table 32 — Two byte extended logical unit addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (00b)		EXTENDED ADDRESS METHOD			
n+1	EXTENDED ADDRESS METHOD SPECIFIC							

Table 33 — Four byte extended logical unit addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (01b)		EXTENDED ADDRESS METHOD			
n+1	EXTENDED ADDRESS METHOD SPECIFIC							
•••								
n+3								

Table 34 — Six byte extended logical unit addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (10b)		EXTENDED ADDRESS METHOD			
n+1	EXTENDED ADDRESS METHOD SPECIFIC							
•••								
n+5								

Table 35 — Eight byte extended logical unit addressing format

Bit Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (11b)		LENGTH (11b)		EXTENDED ADDRESS METHOD			
1	EXTENDED ADDRESS METHOD SPECIFIC							
•••								
7								

The EXTENDED ADDRESS METHOD field combined with the LENGTH field (see table 36) specifies the type and size of extended logical unit address found in the EXTENDED ADDRESS METHOD SPECIFIC field.

Table 36 — Logical unit extended addressing

EXTENDED ADDRESS METHOD Codes	LENGTH Code(s)	Description	Hierarchical LUN usage [a]	Logical unit conglomerate LUN usage [b]	Reference
0h	00b to 11b	Reserved			
1h	00b	Well known logical unit	yes	no	4.7.7.5.1
	01b to 11b	Reserved			
2h	01b	Extended flat space addressing	yes	yes	4.7.7.5.2
	10b	Long extended flat space addressing	yes	yes	4.7.7.5.3
	00b, 11b	Reserved			
3h to Dh	00b to 11b	Reserved			
Eh	00b to 10b	Reserved			
	11b	Restricted for FC-SB-5	no	no	FC-SB-5
Fh	00b to 10b	Reserved			
	11b	Logical unit not specified	yes	no	4.7.7.5.4

[a] Key:
    yes = Other subclauses in this standard are allowed the use of a specific logical unit extended addressing method in hierarchical LUN structures (see 4.7.6.3).
    no = Only a single level LUN structure allowed (see 4.7.5).
[b] Key:
    yes = Other subclauses in this standard are allowed the use of a specific logical unit extended addressing method in logical unit conglomerate LUN structures (see 4.7.6.2).
    no = Other subclauses in this standard are not allowed the use of a specific logical unit extended addressing method in logical unit conglomerate LUN structures.

4.7.7.5.1 Well known logical unit addressing

A SCSI target device may support zero or more well known logical units (see 4.6.17). A single SCSI target device shall only support one instance of each supported well known logical unit. All well known logical units within a SCSI target device shall be accessible from all SCSI target ports contained within the SCSI target device.

Well known logical units are addressed using the well known logical unit extended address format (see table 37).

Table 37 — Well known logical unit extended addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (00b)		EXTENDED ADDRESS METHOD (1h)			
n+1	W-LUN							

The W-LUN field specifies the well known logical unit to be addressed (see SPC-4).

4.7.7.5.2 Extended flat space addressing method

The extended flat space addressing method (see table 38) specifies a logical unit at the current level or in a logical unit conglomerate.

The contents of all hierarchical LUN structure addressing fields following an extended flat space addressing method addressing field shall be ignored. The contents of all logical unit conglomerate LUN structure elements following an extended flat space addressing method addressing field shall be processed as described in 4.7.6.2.

Table 38 — Extended flat space addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (01b)		EXTENDED ADDRESS METHOD (2h)			
n+1	(MSB)							
•••			EXTENDED FLAT SPACE LUN					
n+3								(LSB)

The EXTENDED FLAT SPACE LUN field specifies a current level logical unit.

4.7.7.5.3 Long extended flat space addressing method

The long extended flat space addressing method (see table 39) specifies a logical unit at the current level.

The contents of all hierarchical LUN structure addressing fields following a long extended flat space addressing method addressing field shall be ignored.

Table 39 — Long extended flat space addressing format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (10b)		EXTENDED ADDRESS METHOD (2h)			
n+1	(MSB)							
•••			LONG EXTENDED FLAT SPACE LUN					
n+5								(LSB)

The LONG EXTENDED FLAT SPACE LUN field specifies a current level logical unit.

4.7.7.5.4 Logical unit not specified addressing

Logical unit not specified addressing (see table 40) shall be used to indicate that no logical unit of any kind is specified.

The contents of all hierarchical LUN structure addressing fields following a logical unit not specified addressing method addressing field shall be ignored.

Table 40 — Logical unit not specified extended addressing format

Bit Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (11b)		LENGTH (11b)		EXTENDED ADDRESS METHOD (Fh)			
1	FFh							

4.8 SCSI ports

4.8.1 SCSI port configurations

A SCSI device contains only the following combinations of SCSI ports:

    a) all SCSI target ports;
    b) all SCSI initiator ports; or
    c) any combination of SCSI target ports and SCSI initiator ports.

Some of the SCSI port configurations possible for a SCSI device are shown in figure 24.
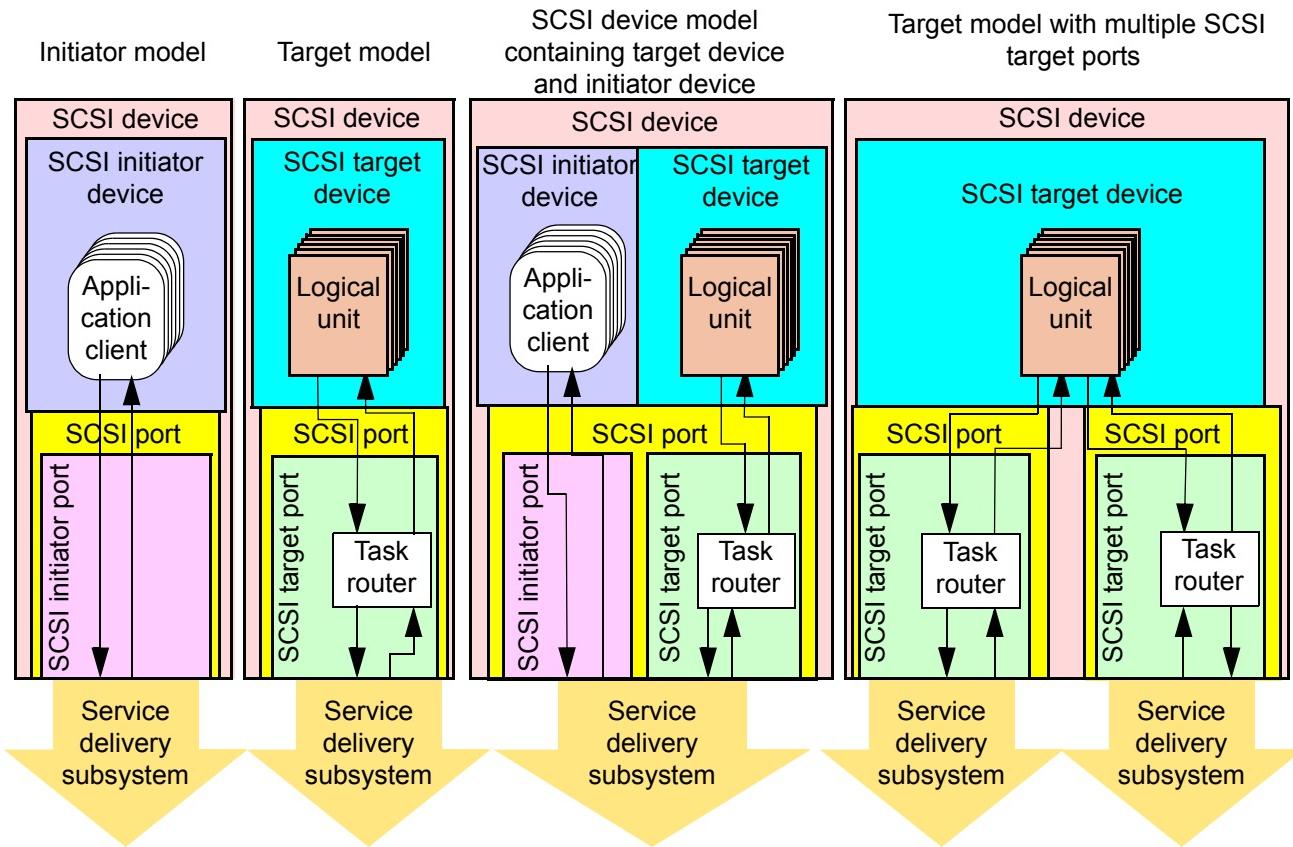
Figure 24 — SCSI device functional models

Additional SCSI initiator ports and SCSI target ports allow the definition of multiple I_T nexuses through which the application client may access the device server.

4.8.2 SCSI devices with multiple SCSI ports

The model for a SCSI device with multiple SCSI ports is a single SCSI device (see 4.6.4) containing either:

a) a SCSI target device (see 4.6.9) and multiple SCSI ports (see 4.6.5) with each SCSI port containing a SCSI target port (see 4.6.6);
b) a SCSI initiator device (see 4.6.27) and multiple SCSI ports, with each SCSI port containing a SCSI initiator port (see 4.6.8); or
c) a SCSI initiator device, a SCSI target device, and multiple SCSI ports with each SCSI port containing a SCSI target port and/or SCSI initiator port.

The identifiers representing the SCSI ports shall meet the requirements for initiator port identifiers (see 4.6.8.2) or target port identifiers (see 4.6.6.2). How a SCSI device with multiple SCSI ports is viewed by counterpart SCSI devices in the SCSI domain also depends on whether a SCSI initiator port is accessing a SCSI target port or a SCSI target port is servicing a SCSI initiator port.

If one SCSI target port is being used by a SCSI initiator port, then accesses attempted through other SCSI target ports may:

a) receive a status of BUSY (see 5.3.1); or
b) be accepted as if the other SCSI target ports were not in use.

4.8.3 SCSI target device with multiple SCSI ports structure

Figure 25 shows the structure of a SCSI target device with multiple SCSI ports each containing a SCSI target port. Each SCSI target port contains a task router that is shared by a collection of logical units. Each logical unit contains a single task manager and a single device server.
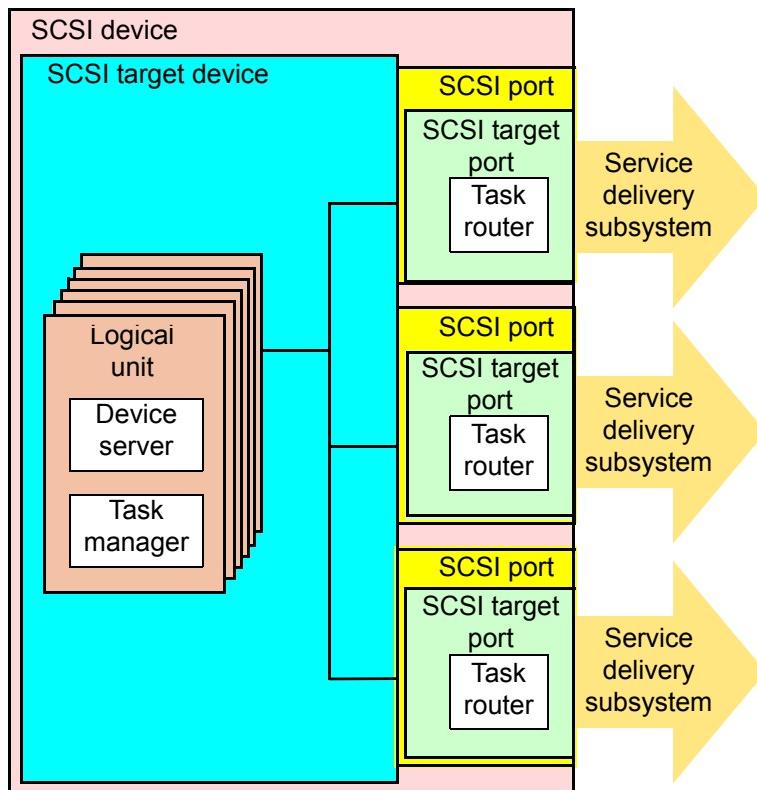
Figure 25 — SCSI device with multiple SCSI target ports structure model

Each device server shall indicate the presence of multiple SCSI target ports by setting the MULTIP bit to one in its standard INQUIRY data (see SPC-4).

Two-way communication is possible between all logical units and all SCSI target ports in a SCSI device. However, communication between any logical unit and any SCSI target port in a SCSI device may be inactive. Two-way communication is possible between each task manager and all task routers in the SCSI target ports in the SCSI device. Each SCSI target port in a SCSI device shall accept commands sent to LUN 0 or the REPORT LUNS well known logical unit, and the task router in that SCSI target port shall route the commands to a device server in a logical unit in the SCSI device for processing. REPORT LUNS commands (see SPC-4) shall be accepted by the logical unit with the LUN 0 or the REPORT LUNS well known logical unit from any SCSI target port in the SCSI device, and the logical unit shall return the logical unit inventory available via that SCSI target port. An application client (e.g., driver) determines the availability of the same logical unit through multiple SCSI target ports in a SCSI device by matching logical unit name values in the Device Identification VPD page (see SPC-4).

4.8.4 SCSI initiator device with multiple SCSI initiator ports structure

Figure 26 shows the structure of a SCSI initiator device with multiple SCSI ports each containing a SCSI initiator port. Each SCSI initiator port is shared by a collection of application clients.
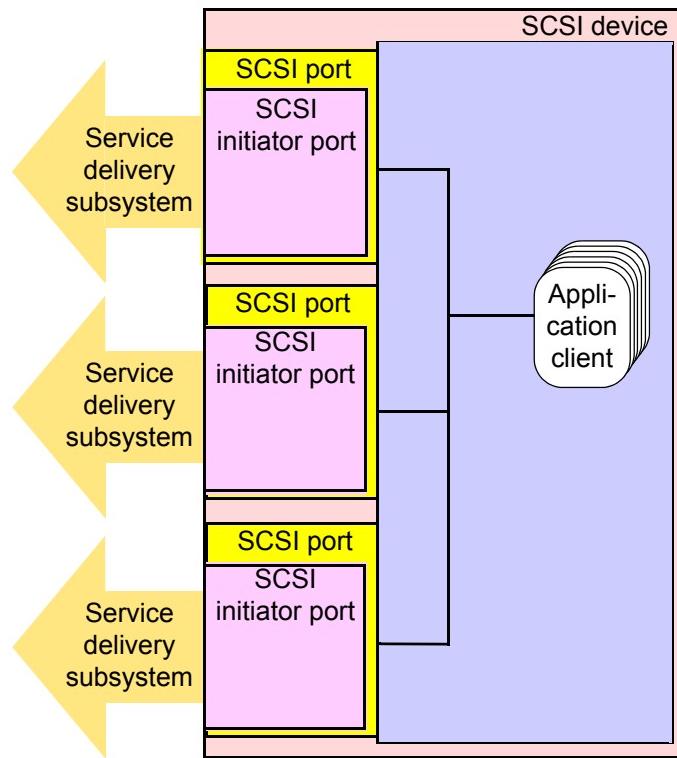
Figure 26 — SCSI initiator device with multiple SCSI initiator ports structure model

Two-way communication is possible between an application client and its associated SCSI initiator ports. This standard does not specify or require the definition of any mechanisms by which a SCSI target device has the ability to discover that it is communicating with multiple SCSI initiator ports on a single SCSI initiator device. In those SCSI transport protocols where such mechanisms are defined, they shall not have any effect on how commands are processed (e.g., reservations shall be handled as if no such mechanisms exist).

4.8.5 SCSI device with multiple SCSI ports structure

Figure 27 shows the structure of a SCSI device containing a SCSI target device and a SCSI initiator device, and multiple SCSI ports. Each SCSI port may contain a SCSI target port and/or a SCSI initiator port. This SCSI device may also contain SCSI ports that only contain a SCSI target port or a SCSI initiator port. Each SCSI port may consist of:

   a) a SCSI target port containing a task router;
   b) a SCSI initiator port; or
   c) a SCSI target port containing a task router and a SCSI initiator port.

The collection of SCSI target ports is shared by a collection of logical units. The collection of SCSI initiators is shared by a collection of application clients.

Each logical unit contains a task manager and a device server.
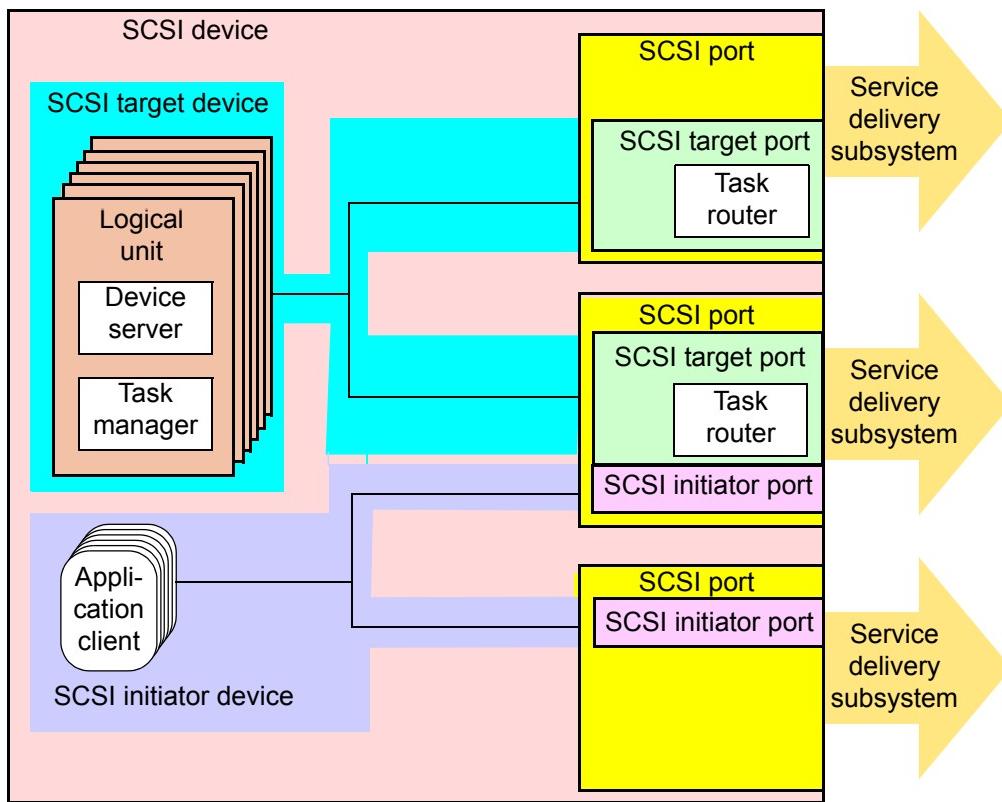
Figure 27 — SCSI device with multiple SCSI ports structure model

Two-way communication is possible between all logical units and all SCSI target ports in a SCSI device. However, communication between any logical unit and any SCSI target port in a SCSI device may be inactive. Two-way communication is possible between each task manager and all task routers in the SCSI target ports in the SCSI device. Each SCSI target port in a SCSI device shall accept commands sent to LUN 0 or the REPORT LUNS well known logical unit, and the task router in that SCSI target port shall route the commands to a device server in a logical unit in the SCSI device for processing. REPORT LUNS commands (see SPC-4) shall be accepted by the logical unit with the LUN 0 or the REPORT LUNS well known logical unit from any SCSI target port in the SCSI device, and the logical unit shall return the logical unit inventory available via that SCSI target port. An application client determines the availability of the same logical unit through multiple SCSI target ports in a SCSI device by matching logical unit name values in the Device Identification VPD page (see SPC-4).

This standard does not specify or require the definition of any mechanisms by which a SCSI target device has the ability to discover that it is communicating with multiple SCSI ports that also contain a SCSI initiator port on a single SCSI device. In those SCSI transport protocols where such mechanisms are defined, they shall not have any effect on how commands are processed (e.g., reservations shall be handled as if no such mechanisms exist).

4.8.6 SCSI initiator device view of SCSI target device with multiple SCSI target ports

A SCSI target device may have SCSI target ports connected to different SCSI domains such that a SCSI initiator port is only able to communicate with the logical units in the SCSI target device using the SCSI target ports in a single SCSI domain. However, SCSI target devices with multiple SCSI ports may be configured where application clients have the ability to discover that one or more logical units are accessible via multiple SCSI target ports. Figure 28 and figure 29 show two examples of such configurations.

Figure 28 shows a SCSI target device with multiple SCSI ports each containing a SCSI target port participating in a single SCSI domain with two SCSI initiator devices. There are three SCSI devices, one of which has two SCSI target ports, and two of which have one SCSI initiator port each. There are two target port identifiers and two initiator port identifiers in this SCSI domain. Using the INQUIRY command's Device Identification VPD page (see SPC-4), the application clients in each of the SCSI initiator devices have the ability to discover if the logical units in the SCSI target devices are accessible via multiple SCSI target ports and map the configuration of the SCSI target device.
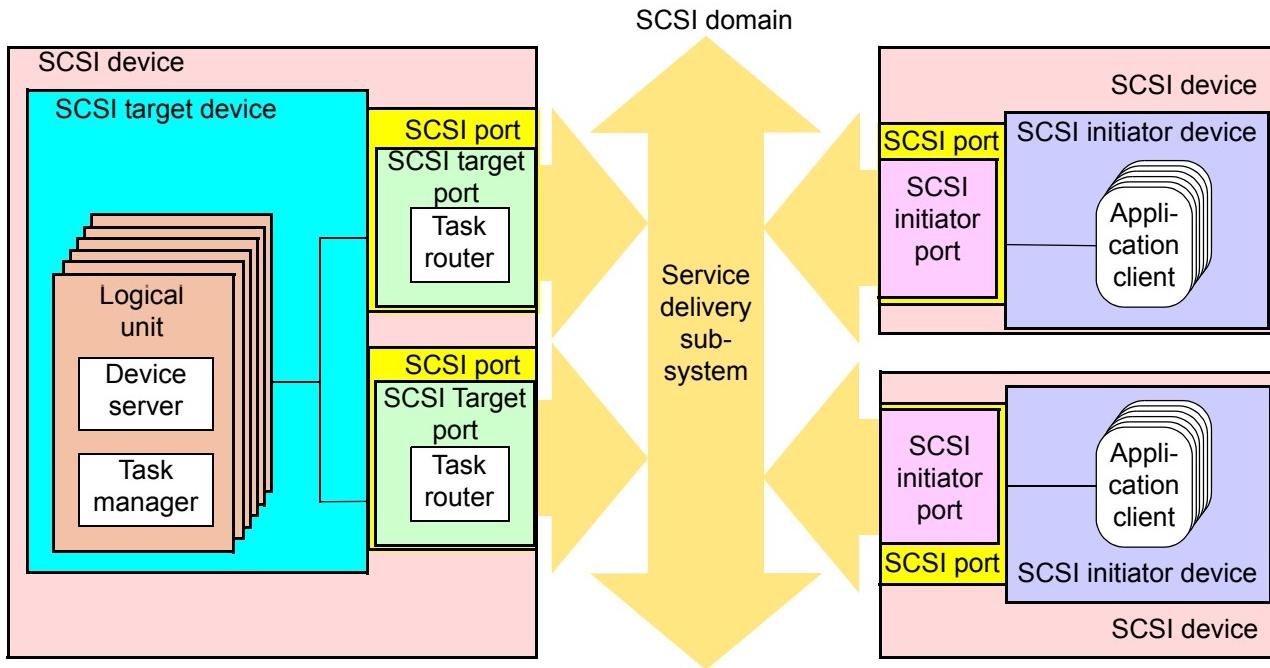
Figure 28 — SCSI target device configured in a single SCSI domain

Figure 29 shows a SCSI target device with multiple SCSI ports each containing a SCSI target port participating in two SCSI domains and a SCSI initiator device with multiple SCSI ports each containing a SCSI initiator port participating in the same two SCSI domains. There is one SCSI target device with two SCSI target ports and one SCSI initiator device with two SCSI initiator ports. There is one target port identifier and one initiator port identifier in each of the two SCSI domains. Using the INQUIRY command's Device Identification VPD page (see SPC-4), the application clients in the SCSI initiator device have the ability to discover that logical units in the SCSI target device are accessible via multiple SCSI initiator ports and multiple SCSI target ports and map the configuration. However, application clients may not be able to distinguish between the configuration shown in figure 29 and the configuration shown in figure 30.
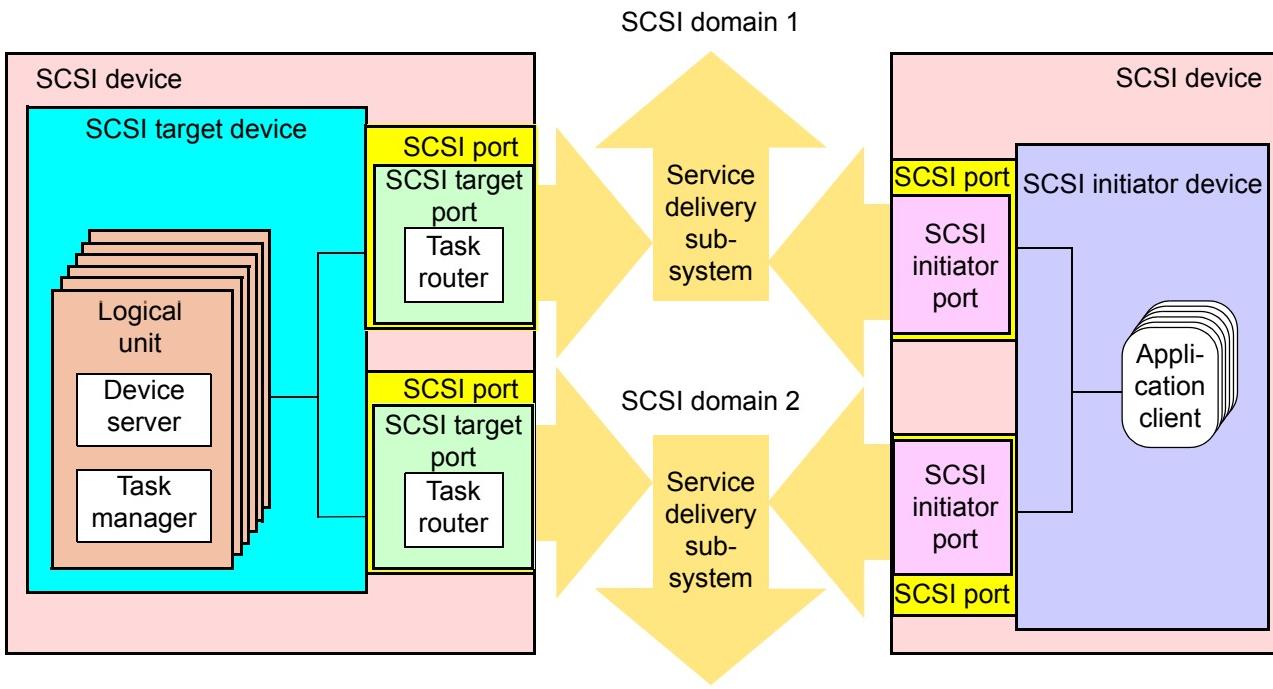
Figure 29 — SCSI target device configured in multiple SCSI domains

Figure 30 shows the same configuration as figure 29 except that the two SCSI domains have been replaced by a single SCSI domain.
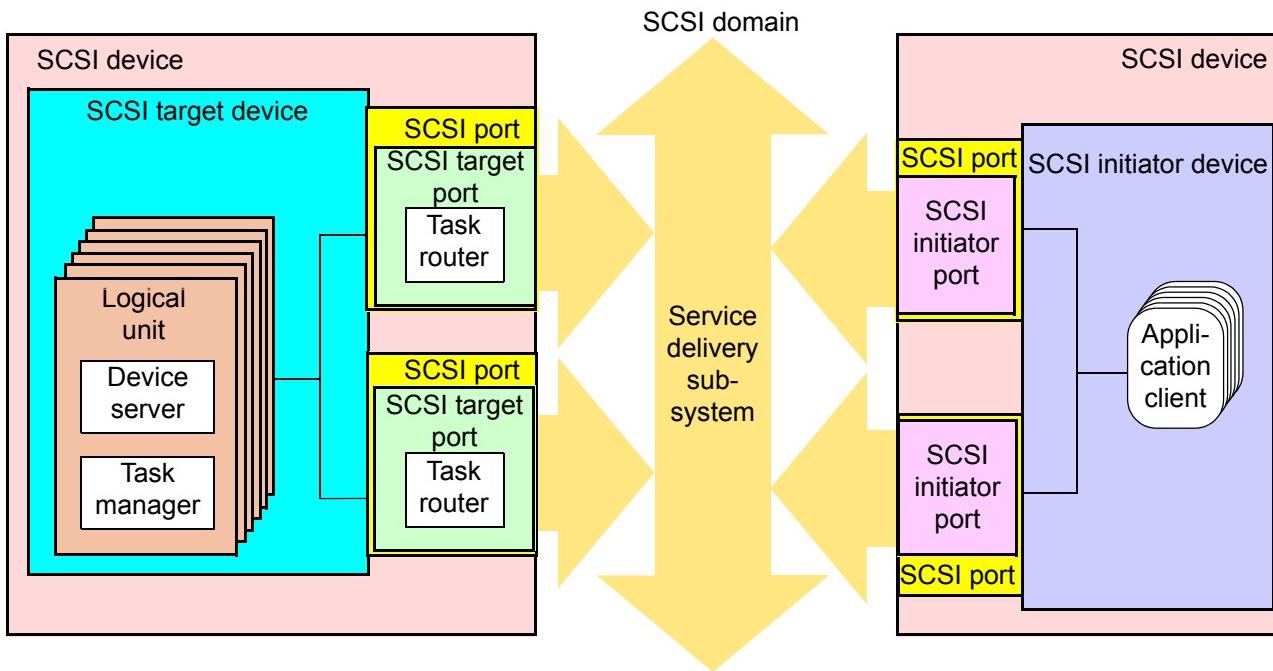
Figure 30 — SCSI target device and SCSI initiator device configured in a single SCSI domain

This model for application client determination of multiple SCSI target port configurations relies on information that is available to the application clients only via commands.

4.8.7 SCSI target device view of a SCSI initiator device with multiple SCSI initiator ports

This standard does not require a SCSI target device to be able to detect that a SCSI initiator device contains more than one SCSI initiator port. In the cases where a SCSI target device does not detect that a SCSI initiator device contains more than one SCSI initiator port, the SCSI target device interacts with the SCSI initiator device as if each SCSI initiator port was contained in a separate SCSI initiator device (e.g., a SCSI target device operates in the configurations shown in figure 29 and figure 30 in the same way it operates in the configuration shown in figure 28).

> NOTE 4 - The implications of this view of a SCSI initiator device are more extensive than are immediately apparent (e.g., after a SCSI initiator device makes a persistent exclusive access reservation via one SCSI initiator port, access is denied to the other SCSI initiator port(s) on that same SCSI initiator device).

4.9 The SCSI model for distributed communications

The SCSI model for communications between distributed objects is based on the technique of layering as shown in figure 31.
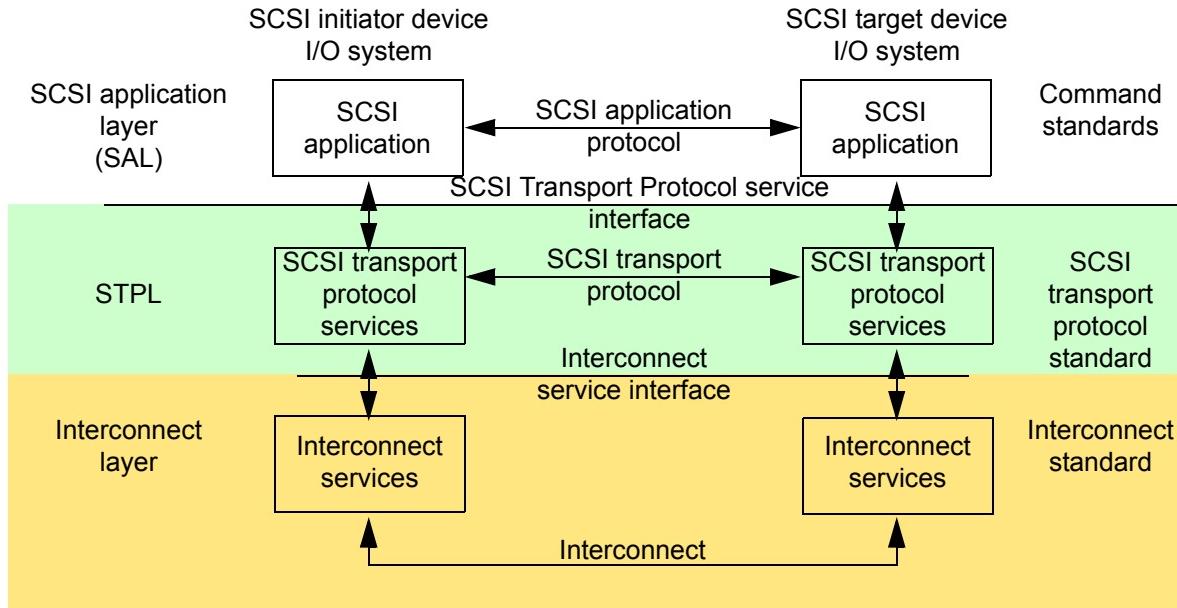
Figure 31 — Protocol service reference model

The layers in this model and the specifications defining the functionality of each layer are denoted by horizontal sequences. A layer consists of entities within the same layer that communicate with one another by means of a protocol. Except for the interconnect layer, such communication is accomplished by invoking services provided by the adjacent layer. The following layers are defined:

a) **SAL:** Clients and servers that originate and process SCSI I/O operations by means of a SCSI application protocol;
b) **STPL:** Services and protocols through which clients and servers communicate; and
c) **Interconnect layer:** Services, signaling mechanism and interconnect subsystem used for the physical transfer of data from sender to receiver. In the SCSI model, the interconnect layer is known as a service delivery subsystem.

The set of SCSI transport protocol services implemented by a service delivery subsystem identify external behavioral requirements that apply to SCSI transport protocol standards. While these SCSI transport protocol services may serve as a guide for designing reusable software or firmware that is adaptable to different SCSI transport protocols, there is no requirement for an implementation to provide the service interfaces defined in this standard.

The SCSI transport protocol service interface is defined in this standard in representational terms using SCSI transport protocol services. The SCSI transport protocol service interface implementation is defined in each SCSI transport protocol standard. The interconnect service interface is described as appropriate in each SCSI transport protocol standard.

Interactions between the SAL and STPL are defined with respect to the SAL and may originate in either layer. An outgoing interaction is modeled as a procedure call invoking an STPL service (e.g., invoking an operation defined by the SCSI Target Port class or the SCSI Initiator Port class). An incoming interaction is modeled as a procedure call invoked by the STPL (e.g., invoking an operation defined by the Application Client class, the Device Server class, or the Task Manager class).

All procedure calls may be accompanied by parameters or data. Both types of interaction are described using the notation for procedures described in 3.9. In this standard, input arguments are defined relative to the layer receiving an interaction (i.e., an input is an argument supplied to the receiving layer by the layer initiating the interaction).

The following types of service interactions between layers are defined:

    a) SCSI transport protocol service request procedure calls from the SAL invoking a service provided by the STPL;

    b) SCSI transport protocol service indication procedure calls from the STPL informing the SAL that an asynchronous event has occurred (e.g., the receipt of a peer-to-peer protocol transaction);

    c) SCSI transport protocol service response procedure calls to the STPL invoked by the SAL in response to a SCSI transport protocol service indication. A SCSI transport protocol service response may be invoked to return a reply from the invoking SAL to the peer SAL; and

    d) SCSI transport protocol service confirmation procedure calls from the STPL notifying the SAL that a SCSI transport protocol service request has completed, has been terminated, or has failed to transit the interconnect layer. A SCSI transport protocol service confirmation may communicate parameters that indicate the completion status of the SCSI transport protocol service request or any other status. A SCSI transport protocol service confirmation may be used to convey a response from the peer SAL.

The services provided by an STPL are either confirmed or unconfirmed (i.e., service available at the SCSI transport protocol service interface that does not result in a completion confirmation). A SAL service request invoking a confirmed service always results in a confirmation from the STPL.

Figure 32 shows the relationships between the four SCSI transport protocol service types.
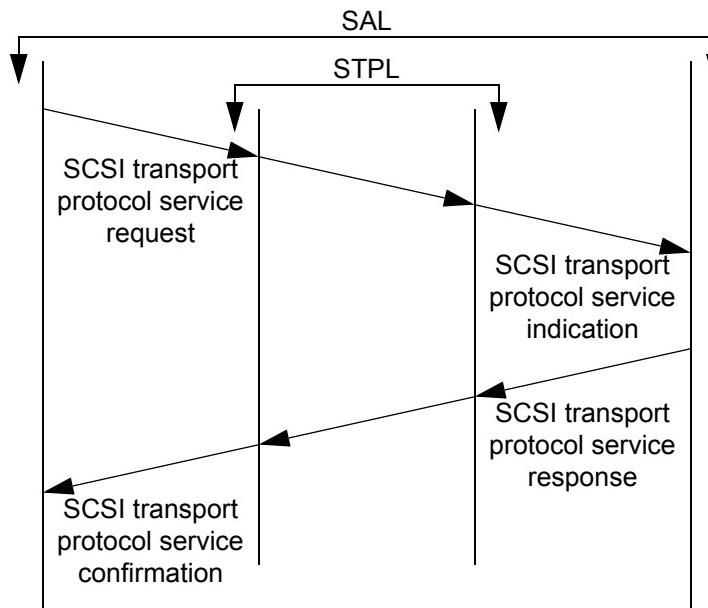
Figure 32 — SCSI transport protocol service mode

Figure 33 shows how SCSI transport protocol services may be used to process a client-server request-response transaction at the SAL.
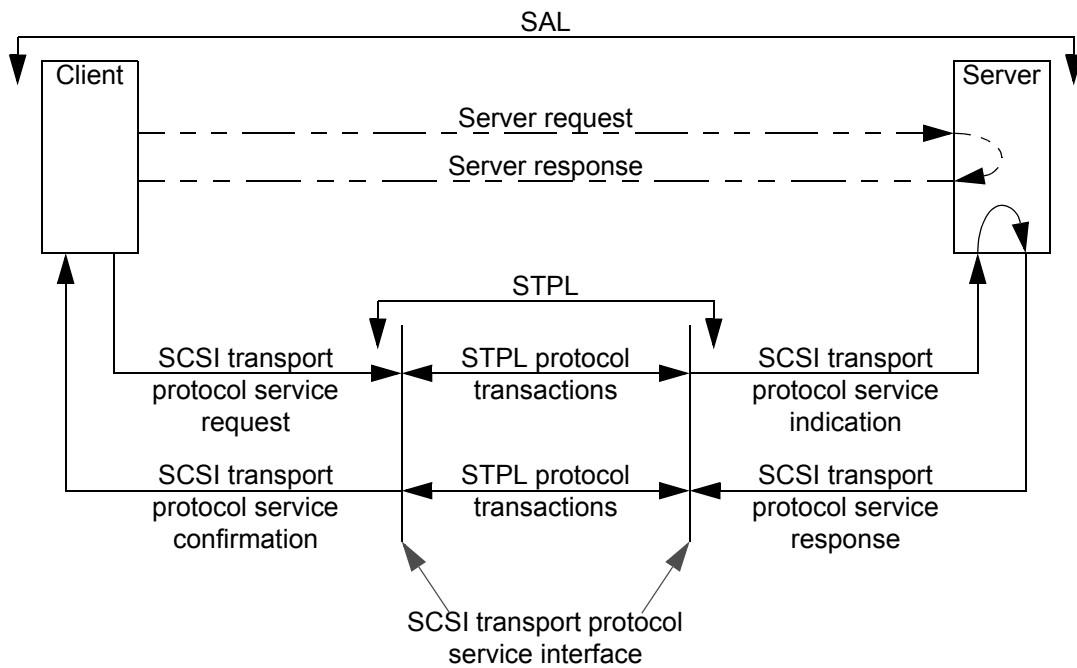
Figure 33 — Request-Response SAL transaction and related STPL services

The dashed lines in figure 33 show a SCSI application protocol transaction as it may appear to sending and receiving entities within the client and server. The solid lines in figure 33 show the corresponding SCSI transport protocol services and STPL transactions that are used to transport the data.

If a device server invokes a data transfer SCSI transport protocol service, then the interactions required to transfer the data do not involve the application client. Only the STPL in the SCSI device that also contains the application client is involved. Figure 34 shows the relationships between the SCSI transport protocol service types involved in a data transfer request.
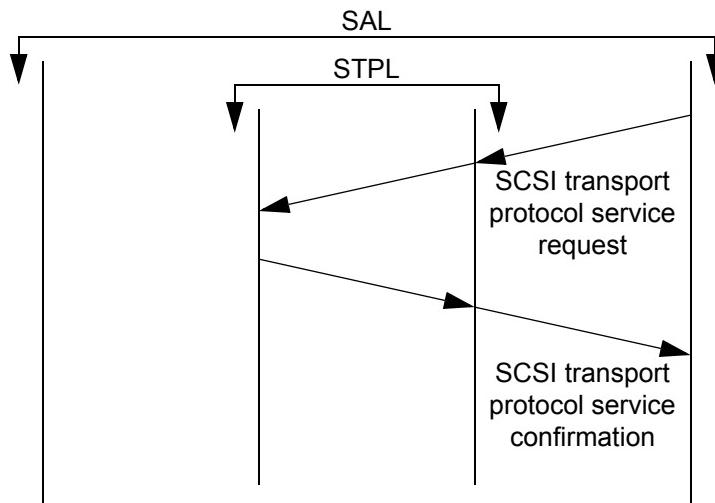
Figure 34 — SCSI transport protocol service model for data transfers

Figure 35 shows how SCSI transport protocol services may be used to process a device server data transfer transaction.
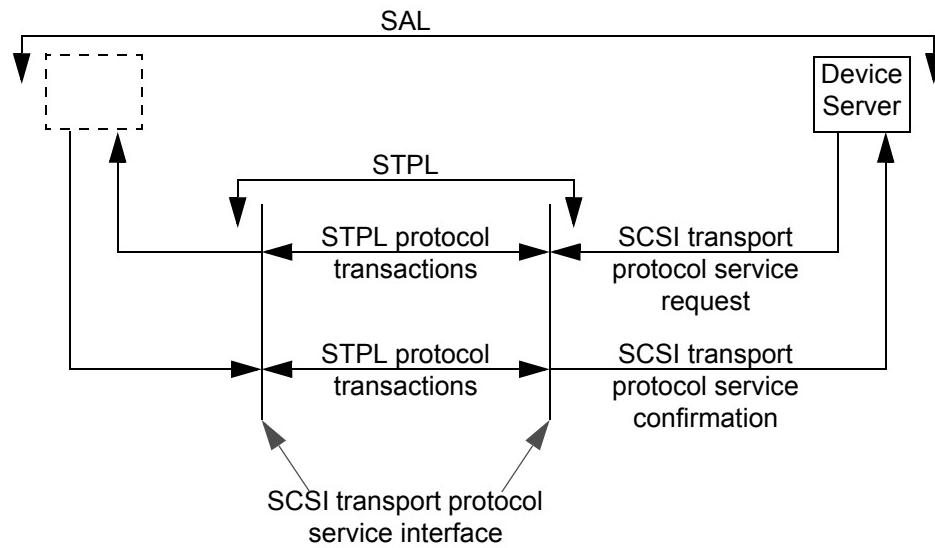
Note: The dotted box represents a memory access function provided by the
SCSI initiator device whose definition is outside the scope of this standard.

Figure 35 — Device server data transfer transaction and related STPL services

If a device server invokes a Terminate Data Transfer SCSI transport protocol service, then the interactions required to complete the service do not involve the SCSI transport protocol service interface or the application client. Only the STPL in the SCSI device that also contains the device server is involved. Figure 36 shows the relationships between the SCSI transport protocol service types involved in a Terminate Data Transfer request.
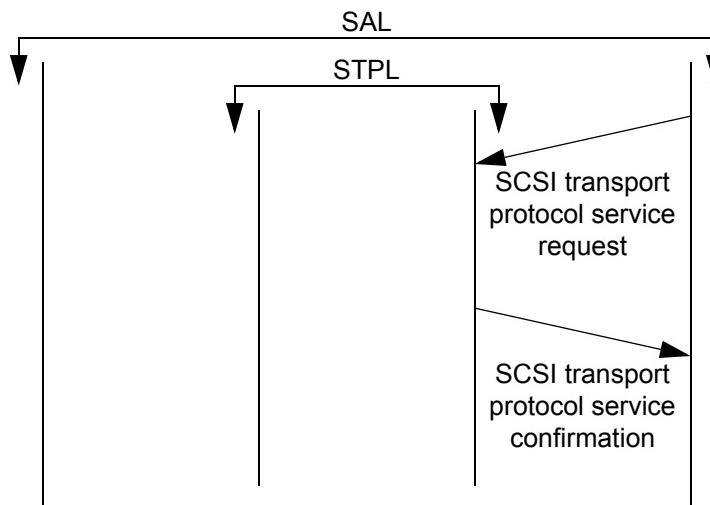
Figure 36 — SCSI transport protocol service model for Terminate Data Transfer

Figure 37 shows how SCSI transport protocol services may be used to process a device server Terminate Data Transfer transaction.

.
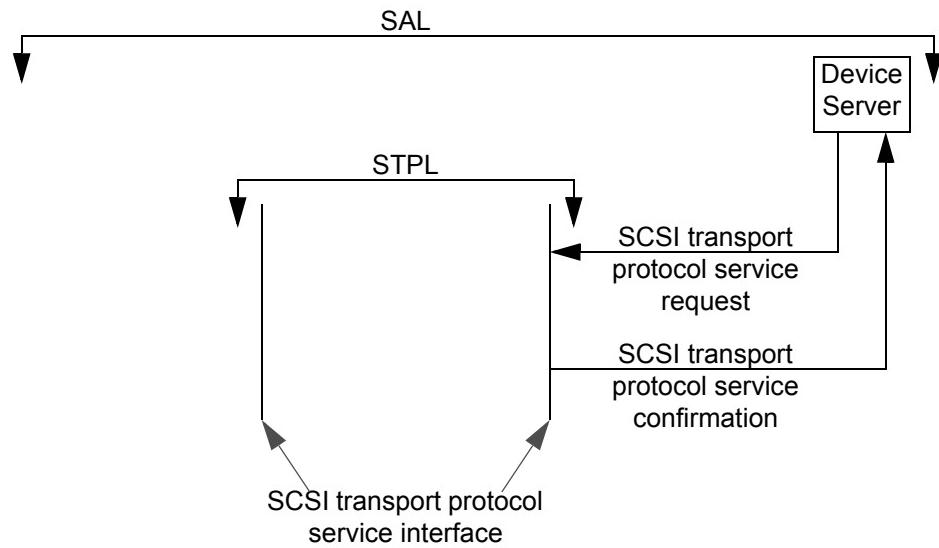
Figure 37 — Device server Terminate Data Transfer transaction and related STPL services

5 SCSI command model

5.1 The Execute Command procedure call

An application client requests the processing of a command by invoking the SCSI transport protocol services described in 5.4, the collective operation of which is modeled in the following procedure call:

Service Response =   Execute Command   (IN (I_T_L Nexus, Command Identifier, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [CRN], [Command Priority]), OUT ([Data-In Buffer], [Sense Data], [Sense Data Length], Status, [Status Qualifier]))

Input arguments:

I_T_L Nexus: An identifier for the I_T_L nexus routing the command (see 4.6.31.2).

Command Identifier: The numerical identifier identifying the command (see 4.6.31.3).

CDB: Command descriptor block (see 5.2).

Task Attribute: A value specifying one of the task attributes defined in 8.4.

Data-In Buffer Size: The number of bytes available for data transfers to the Data-In Buffer (see 5.4.3). SCSI transport protocols may interpret the Data-In Buffer Size to include both the size and the location of the Data-In Buffer.

Data-Out Buffer: A buffer (see 5.4.3) containing command specific information to be sent to the logical unit (e.g., data or parameter lists needed to process the command). The buffer size is indicated by the Data-Out Buffer Size argument. The content of the buffer shall not change during the lifetime of the command (see 5.5) as viewed by the application client.

Data-Out Buffer Size: The number of bytes available for data transfers from the Data-Out Buffer (see 5.4.3).

CRN: If the CRN is used, then all commands on an I_T_L nexus shall include a CRN argument that is incremented by one after each Send SCSI Command operation (see 4.6.8.4) is invoked. The CRN shall be set to one for each I_T_L nexus involving the SCSI port after the SCSI port receives a hard reset or detects I_T nexus loss. The CRN shall be set to one after it reaches the maximum CRN value supported by the protocol. The CRN value zero shall be reserved for use as defined by the SCSI transport protocol. It is not an error for the application client to provide a CRN when CRN is not supported by the SCSI transport protocol or logical unit. See the SCSI transport protocol standards for rules regarding CRN checking.

Command Priority: The priority assigned to the command (see 8.5).

Output arguments:

Data-In Buffer: A buffer (see 5.4.3) to contain command specific information returned by the logical unit by the time of command completion. The **Execute Command** procedure call shall not return GOOD status or CONDITION MET status unless the buffer contents are valid. The buffer contents shall be considered invalid unless the **Execute Command** procedure call returns GOOD status or CONDITION MET status. While some valid data may be present for other values of status, the application client should rely on additional information from the logical unit (e.g., sense data) to determine the state of the buffer contents. If the command terminates with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider the buffer contents to be undefined.

Sense Data: A buffer containing sense data (see 5.13) returned in the same I_T nexus transaction as status (see 5.3). The buffer length is indicated by the Sense Data Length argument. If the command terminates with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider the sense data to be undefined.

Sense Data Length: The length in bytes of the Sense Data (see 5.13).

Status: A one-byte field containing command completion status (see 5.3). If the command terminates with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider command completion status to be undefined.

Status Qualifier: Additional information about the indicated command completion status (see 5.3.2).

One of the following SCSI transport protocol specific service responses shall be returned:

COMMAND COMPLETE: A logical unit response indicating that the command has completed. The Status argument shall have one of the values described in 5.3.

SERVICE DELIVERY OR TARGET FAILURE: The command has been terminated due to a service delivery failure or SCSI target device malfunction. All output arguments are invalid.

The SCSI transport protocol events corresponding to a service response of COMMAND COMPLETE or SERVICE DELIVERY OR TARGET FAILURE shall be specified in each SCSI transport protocol standard.

5.2 Command descriptor block (CDB)

The CDB defines the operation to be performed by the device server. See SPC-4 for the CDB formats.

For all commands, if the logical unit detects an invalid field in the CDB, then the logical unit shall not process the command.

All CDBs shall have an OPERATION CODE field as the first byte.

Some operation codes provide for modification of their operation based on a service action. In such cases, the combination of operation code value and service action code value may be modeled as a single, unique command. The location of the SERVICE ACTION field in the CDB varies depending on the operation code value.

All CDBs shall contain a CONTROL byte (see table 41). The location of the CONTROL byte within a CDB depends on the CDB format (see SPC-4).

Table 41 — CONTROL byte

Bit	7	6	5	4	3	2	1	0
	Vendor specific		Reserved			NACA	Obsolete	Obsolete

All SCSI transport protocol standards shall define as mandatory the functionality needed for a logical unit to implement the NACA bit.

The NACA (Normal ACA) bit specifies whether an ACA condition is established if the command terminates with CHECK CONDITION status. A NACA bit set to one specifies that an ACA shall be established. A NACA bit set to zero specifies that an ACA shall not be established. The actions for ACA are described in 5.9. Actions that may be required when an ACA is not established are described in 5.8. All logical units shall implement support for the NACA value of zero and may support the NACA value of one (i.e., ACA). The ability to support a NACA value of one is indicated with the NORMACA bit in the standard INQUIRY data (see SPC-4).

If the NACA bit is set to one but the logical unit does not support ACA, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

5.3 Status

5.3.1 Status codes

The status codes are described in table 42. Status shall be sent from the device server to the application client whenever a command completes with a service response of COMMAND COMPLETE.

Table 42 — Status codes

Code	Status	Command completed	Service response
00h	GOOD	yes	COMMAND COMPLETE
02h	CHECK CONDITION	yes	COMMAND COMPLETE
04h	CONDITION MET	yes	COMMAND COMPLETE
08h	BUSY	yes	COMMAND COMPLETE
10h	Obsolete		
14h	Obsolete		
18h	RESERVATION CONFLICT	yes	COMMAND COMPLETE
22h	Obsolete		
28h	TASK SET FULL	yes	COMMAND COMPLETE
30h	ACA ACTIVE	yes	COMMAND COMPLETE
40h	TASK ABORTED	yes	COMMAND COMPLETE
All other codes	Reserved		

Sense data may be delivered in the buffer defined by the Sense Data argument of the **Execute Command** procedure call (see 5.1) for any status code.

Definitions for each status code are as follows:

GOOD: This status indicates that the device server has completed the command without error.

CHECK CONDITION: This status indicates that sense data has been delivered in the buffer defined by the Sense Data argument to the **Execute Command** procedure call (see 5.13). Additional actions that are required if CHECK CONDITION status is returned are described in 5.8.

CONDITION MET: The use of this status is limited to commands for which it is specified (see the PRE-FETCH commands in SBC-3).

BUSY: This status indicates that the logical unit is busy. This status shall be returned whenever a logical unit is temporarily unable to accept a command through the SCSI target port on which the status is returned and zero or more other SCSI target ports. The recommended application client recovery action is to issue the command again at a later time.

If the UA_INTLCK_CTRL field in the Control mode page contains 11b (see SPC-4), termination of a command with BUSY status shall cause a unit attention condition to be established for the SCSI initiator port on the I_T nexus that sent the command with an additional sense code set to PREVIOUS BUSY STATUS.

The status qualifier, if supported by a SCSI transport protocol, may provide the SCSI initiator port with more information about when the command should be retransmitted (see 5.3.2).

RESERVATION CONFLICT: This status shall be returned whenever a command is sent by an application client to a logical unit in a way that conflicts with an existing reservation (see SPC-4.)

If the UA_INTLCK_CTRL field in the Control mode page contains 11b (see SPC-4), termination of a command with RESERVATION CONFLICT status shall cause a unit attention condition to be established for the SCSI initiator port on the I_T nexus that sent the command with an additional sense code set to PREVIOUS RESERVATION CONFLICT STATUS.

TASK SET FULL: If the logical unit has at least one command in the task set for an I_T nexus and a lack of task set resources prevents the logical unit from accepting an additional command received from that I_T nexus into the task set, then TASK SET FULL status shall be returned. If the logical unit has no command in the task set for an I_T nexus and a lack of task set resources prevents accepting a received command from that I_T nexus into the task set, then BUSY status should be returned.

The logical unit should allow at least one command in the task set for each supported I_T nexus (i.e., a logical unit should allow at least one command into the task set for each I_T nexus that has been identified in a SCSI transport protocol specific manner (e.g., a login), or by the successful reception of a command).

The status qualifier, if supported by a SCSI transport protocol, may provide the SCSI initiator port with more information about when the command should be retransmitted (see 5.3.2).

If the UA_INTLCK_CTRL field in the Control mode page contains 11b (see SPC-4), termination of a command with TASK SET FULL status shall cause a unit attention condition to be established for the SCSI initiator port on the I_T nexus that sent the command with an additional sense code set to PREVIOUS TASK SET FULL STATUS.

ACA ACTIVE: This status shall be returned as described in 5.9.2 and 5.9.3 if an ACA exists within a task set. The application client may reissue the command on the same I_T nexus after the ACA condition has been cleared.

TASK ABORTED: This status shall be returned if a command is aborted by a command or task management function on another I_T nexus and the Control mode page TAS bit is set to one (see 5.6).

5.3.2 Status qualifier

The status qualifier provides additional information about the reason for the status code (see 5.3.1).

The status qualifier format is as shown in table 43.

Table 43 — Status qualifier format

Bit Byte	7	6	5	4	3	2	1	0
0	SCOPE		(MSB)	QUALIFIER				
1								(LSB)

The SCOPE field (see table 44) indicates the logical unit(s) to which the status qualifier applies.

Table 44 — SCOPE field

Code	Affected logical unit(s)	Affected nexus(es)
00b	The logical unit addressed by the command associated with the status.	All I_T nexus(es) associated with the addressed logical unit.
01b	All logical units accessible by the SCSI target port through which the command associated with the status was routed.	I_T nexus through which the status was returned.
10b	All logical unit(s) contained within the SCSI device that contains the logical unit addressed by the command associated with the status.	All I_T nexus(es).
11b	Reserved	

The QUALIFIER field (see table 45) indicates additional information about the reason for the status code.

Table 45 — QUALIFIER field

Status code	QUALIFIER field	Description
All	0000h	No additional information (i.e., the same as returning no status qualifier).
BUSY	0001h to 3FEFh	The number of 100 ms increments the application client should wait before sending another command to the logical unit(s) indicated by the SCOPE field using the nexus(es) indicated by the SCOPE field.
	3FF0h to 3FFDh	Reserved
	3FFEh	The application client should stop sending commands to the logical unit(s) indicated by the SCOPE field using the nexus(es) indicated by the SCOPE field.
	3FFFh	The logical unit(s) indicated by the SCOPE field are not able to accept the command because they are servicing too many other I_T nexuses.
TASK SET FULL [a]	0001h to 3FEFh	The application client should wait before sending another command to the logical unit on any I_T nexus until: a) at least the number of 100 ms increments indicated in the QUALIFIER field have elapsed; or b) a command addressed to the logical unit on any I_T nexus completes or terminates.
	3FF0h to 3FFFh	Reserved
GOOD	0001h to 3FFFh	Reserved
CHECK CONDITION	0001h to 3FEFh	The number of 100 ms increments the application client should wait before sending another command to the logical unit(s) indicated by the SCOPE field using the nexus(es) indicated by the SCOPE field.
	3FF0h to 3FFFh	Reserved
CONDITION MET	0001h to 3FFFh	Reserved
RESERVATION CONFLICT	0001h to 3FFFh	Reserved
ACA ACTIVE	0001h to 3FFFh	Reserved
TASK ABORTED	0001h to 3FFFh	Reserved
All others	0001h to 3FFFh	Reserved
[a] The SCOPE field shall be set to zero.		

5.3.3 Status precedence

If a device server or task manager detects that more than one of the following conditions applies to a completed command, it shall select the condition to report based on the following precedence:

1) an ACA ACTIVE status;
2) a CHECK CONDITION status for any of the following unit attention conditions (i.e., with a sense key set to UNIT ATTENTION and one of the following additional sense codes):
    A) POWER ON, RESET, OR BUS DEVICE RESET OCCURRED;
    B) POWER ON OCCURRED;
    C) SCSI BUS RESET OCCURRED;
    D) MICROCODE HAS BEEN CHANGED;
    E) BUS DEVICE RESET FUNCTION OCCURRED;
    F) DEVICE INTERNAL RESET;
    G) COMMANDS CLEARED BY POWER LOSS NOTIFICATION; or
    H) I_T NEXUS LOSS OCCURRED;
3) a RESERVATION CONFLICT status; and
4) a status of:
    A) CHECK CONDITION, other than with a sense key set to ILLEGAL REQUEST or for any reason not listed in 2);
    B) GOOD;
    C) CONDITION MET; or
    D) TASK ABORTED.

> NOTE 5 - The names of the unit attention conditions listed in this subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in SAM-2. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

A device server or task manager may report the following status codes with any level of precedence:

a) BUSY status;
b) TASK SET FULL status; or
c) CHECK CONDITION status with a sense key set to ILLEGAL REQUEST.

For an administrative logical unit (see 4.6.11) or a hierarchical logical unit (see 4.6.15), any unit attention condition that was established for all logical units (e.g., REPORTED LUNS DATA HAS CHANGED) should be reported with a higher precedence than a CHECK CONDITION status with a sense key set to ILLEGAL REQUEST and an additional sense code set to LOGICAL UNIT NOT SUPPORTED.

For a subsidiary logical unit, any unit attention condition that was established for all logical units in the logical unit conglomerate (e.g., REPORTED LUNS DATA HAS CHANGED) shall be reported with a higher precedence than a CHECK CONDITION status with a sense key set to ILLEGAL REQUEST and an additional sense code set to SUBSIDIARY LOGICAL UNIT NOT CONFIGURED.

5.4 SCSI transport protocol services for SCSI commands

5.4.1 SCSI transport protocol services for SCSI commands overview

The SCSI transport protocol services that support the **Execute Command** procedure call are described in 5.4. The following groups of SCSI transport protocol services are described:

a) the SCSI transport protocol services that support the delivery of the command and status (see 5.4.2); and
b) the SCSI transport protocol services that support the data transfers associated with processing a command (see 5.4.3).

5.4.2 Command and status SCSI transport protocol services

5.4.2.1 Command and status SCSI transport protocol services overview

All SCSI transport protocol standards shall define the SCSI transport protocol specific requirements for implementing the **Send SCSI Command** SCSI transport protocol service request (see 5.4.2.2), the **SCSI Command Received** SCSI transport protocol service indication (see 5.4.2.3), the **Send Command Complete** SCSI transport protocol service response (see 5.4.2.4), and the **Command Complete Received** SCSI transport protocol service confirmation (see 5.4.2.5) SCSI transport protocol services.

All SCSI initiator devices shall implement the **Send SCSI Command** SCSI transport protocol service request and the **Command Complete Received** SCSI transport protocol service confirmation as defined in the applicable SCSI transport protocol standards. All SCSI target devices shall implement the **SCSI Command Received** SCSI transport protocol service indication and the **Send Command Complete** SCSI transport protocol service response as defined in the applicable SCSI transport protocol standards.

5.4.2.2 Send SCSI Command SCSI transport protocol service request

An application client invokes the **Send SCSI Command** SCSI transport protocol service request to request that a SCSI initiator port send a command over the service delivery subsystem.

Send SCSI Command SCSI transport protocol service request:

> **Send SCSI Command (IN (I_T_L Nexus, Command Identifier, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [CRN], [Command Priority], [First Burst Enabled]))**

Input arguments:

I_T_L Nexus:	An identifier for the I_T_L nexus routing the command (see 4.6.32.3).
Command Identifier:	The numerical identifier identifying the command (see 4.6.31.3).
CDB:	Command descriptor block (see 5.2).
Task Attribute:	A value specifying one of the task attributes defined in 8.4. For specific requirements on the Task Attribute argument see 5.1.
Data-In Buffer Size:	The number of bytes available for data transfers to the Data-In Buffer (see 5.4.3). SCSI transport protocols may interpret the Data-In Buffer Size to include both the size and the location of the Data-In Buffer.
Data-Out Buffer:	A buffer containing command specific information to be sent to the logical unit (e.g., data or parameter lists needed to process the command (see 5.1)). The content of the Data-Out Buffer shall not change during the lifetime of the command (see 5.5) as viewed by the application client.
Data-Out Buffer Size:	The number of bytes available for data transfers from the Data-Out Buffer (see 5.4.3).
CRN:	If CRN is used, then all commands on an I_T_L nexus include a CRN argument (see 5.1).
Command Priority:	The priority assigned to the command (see 8.5).
First Burst Enabled:	An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer shall be delivered to the logical unit without waiting for the device server to invoke the **Receive Data-Out** SCSI transport protocol service.

5.4.2.3 SCSI Command Received SCSI transport protocol service indication

The task router (see 4.6.7) invokes the **SCSI Command Received** SCSI transport protocol service indication to notify a task manager that it has received a command over the service delivery subsystem.

SCSI Command Received SCSI transport protocol service indication:

SCSI Command Received **(IN (I_T_L Nexus, Command Identifier, CDB, Task Attribute, [CRN], [Command Priority], [First Burst Enabled]))**

Input arguments:

I_T_L Nexus:	An identifier for the I_T_L nexus routing the command (see 4.6.32.3).
Command Identifier:	The numerical identifier identifying the command (see 4.6.31.3).
CDB:	Command descriptor block (see 5.2).
Task Attribute:	A value specifying one of the task attributes defined in 8.4.
CRN:	If a CRN argument is used, then all commands on an I_T_L nexus include a CRN argument (see 5.1).
Command Priority:	The priority assigned to the command (see 8.5).
First Burst Enabled:	An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer are being delivered to the logical unit without waiting for the device server to invoke the **Receive Data-Out** SCSI transport protocol service.

5.4.2.4 Send Command Complete SCSI transport protocol service response

A device server, a task manager, or a task router invokes the **Send Command Complete** SCSI transport protocol service response to request that a SCSI target port transmit command complete information over the service delivery subsystem.

Send Command Complete SCSI transport protocol service response**:**

Send Command Complete **(IN (I_T_L Nexus, Command Identifier, [Sense Data], [Sense Data Length], Status, Service Response, [Status Qualifier]))**

Input arguments:

I_T_L Nexus:	An identifier for the I_T_L nexus routing the command (see 4.6.32.3).
Command Identifier:	The numerical identifier identifying the command (see 4.6.31.3).
Sense Data:	If present, a Sense Data argument instructs the SCSI target port to complete with sense data to the SCSI initiator port (see 5.13).
Sense Data Length:	The length in bytes of the sense data to be returned to the SCSI initiator port.
Status:	Command completion status (see 5.1).
Service Response:	Service response information for the command (see 5.1).
Status Qualifier:	The Status Qualifier code for the command (see 5.3.2).

5.4.2.5 Command Complete Received SCSI transport protocol service confirmation

A SCSI initiator port invokes the **Command Complete Received** SCSI transport protocol service confirmation to notify an application client that it has received command complete information.

Command Complete Received SCSI transport protocol service confirmation:

Command Complete Received **(IN (I_T_L Nexus, Command Identifier, [Data-In Buffer], [Sense Data], [Sense Data Length], Status, Service Response, [Status Qualifier]))**

Input arguments:

I_T_L Nexus:	An identifier for the I_T_L nexus routing the command (see 4.6.32.3).
Command Identifier:	The numerical identifier identifying the command (see 4.6.31.3).
Data-In Buffer:	A buffer containing command specific information returned by the logical unit on command completion (see 5.1).
Sense Data:	Sense data returned in the same I_T nexus transaction as status (see 5.13).
Sense Data Length:	The length in bytes of the received sense data.
Status:	Command completion status (see 5.1).
Service Response:	Service response for the command (see 5.1).
Status Qualifier:	The status qualifier for the command (see 5.3.2).

5.4.3 Data transfer SCSI transport protocol services

5.4.3.1 Introduction

The data transfer services described in 5.4.3 provide mechanisms for moving data to and from the SCSI initiator port while processing commands. All SCSI transport protocol standards shall define the protocols required to implement these services.

The application client's Data-In Buffer and/or Data-Out Buffer each appears to the device server as a single, logically contiguous block of memory large enough to hold all the data required by the command (see figure 38). This standard allows either unidirectional or bidirectional data transfer. The processing of a command may require the transfer of data from the application client using the Data-Out Buffer, or to the application client using the Data-In Buffer, or both to and from the application client using both the Data-In Buffer and the Data-Out Buffer.
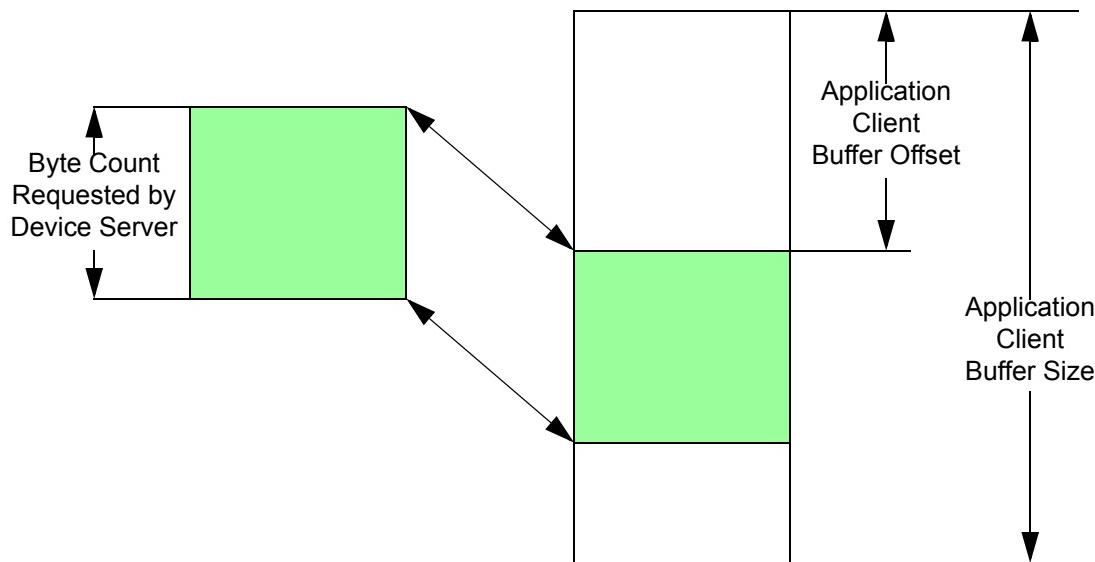
Figure 38 — Model for Data-In and Data-Out data transfers

This standard assumes that the buffering resources available to a logical unit are limited, and the buffer in the logical unit may not be capable of containing all of the data required to be transferred for one command. Such data needs to be moved between the application client and the logical unit in segments that are smaller than the transfer size specified in the command. The amount of data moved per segment is usually a function of the buffering resources available to the logical unit. Figure 38 shows the model for such incremental data transfers.

SCSI transport protocols may allow logical units to accept the initial portion of the Data-Out Buffer data, called the first burst, along with the command without waiting for the device server to invoke the **Receive Data-Out** SCSI transport protocol service. This is modeled using **Receive Data-Out** SCSI transport protocol service calls for which the SCSI transport protocol may have moved the first burst prior to the call.

SCSI transport protocols that define a first burst capability shall include the First Burst Enabled argument in their definitions for the **Send SCSI Command** and **SCSI Command Received** SCSI transport protocol services. Logical units that implement the first burst capability shall implement the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see SPC-4).

The STPL confirmed services described in 5.4.3.2 and 5.4.3.3 are used by the device server to request the transfer of data to or from the application client Data-In Buffer or Data-Out Buffer, respectively. The SCSI initiator device SCSI transport protocol service interactions for data transfers are unspecified.

The movement of data between the application client and device server is controlled by the following arguments:

Application Client Buffer Size: The total number of bytes in the application client's buffer (i.e., equivalent to Data-In Buffer Size for the Data-In Buffer or equivalent to Data-Out Buffer Size for the Data-Out Buffer).

Application Client Buffer Offset: Offset in bytes from the beginning of the application client's buffer (Data-In or Data-Out) to the first byte of transferred data.

Byte Count Requested by Device Server: Number of bytes to be moved by the data transfer request.

For any specific data transfer SCSI transport protocol service request, the **Byte Count Requested by Device Server** is less than or equal to the combination of **Application Client Buffer Size** minus the **Application Client Buffer Offset**.

Random buffer access occurs when the device server requests data transfers to or from segments of the application client's buffer that have an arbitrary offset and byte count. Buffer access is sequential when successive transfers access a series of increasing, adjoining buffer segments. Support for random buffer access by a SCSI transport protocol standard is optional. A device server implementation designed for any SCSI transport protocol implementation should be prepared to use sequential buffer access if necessary.

If a SCSI transport protocol supports random buffer access, the offset and byte count specified for each data segment to be transferred may overlap. In this case the total number of bytes moved for a command is not a reliable indicator of highest byte transferred and shall not be used by a SCSI initiator device or SCSI target device implementation to determine whether all data has been transferred.

All SCSI transport protocol standards shall define support for a resolution of one byte for the Application Client Buffer Size argument.

SCSI transport protocol standards may define restrictions on the resolution of the Application Client Buffer Offset argument. SCSI transport protocol standards may define restrictions on the resolution of the Request Byte Count argument for any call to the **Send Data-In** SCSI transport protocol service or any call to the **Receive Data-Out** SCSI transport protocol service that does not transfer the last byte of the Application Client Buffer.

This standard provides only for the transfer phases to be sequential. Provision for overlapping transfer phases is outside the scope of this standard.

The STPL confirmed services described in 5.4.3.4 are used by the task manager or device server to terminate partially completed transfers to the Data-In Buffer or from the Data-Out Buffer. The **Terminate Data Transfer** SCSI transport protocol service requests that one or more **Send Data-In** SCSI transport protocol service requests or **Receive Data-Out** SCSI transport protocol service requests be terminated by a SCSI target port.

5.4.3.2 Data-In delivery service

5.4.3.2.1 Send Data-In SCSI transport protocol service request

A device server invokes the **Send Data-In** SCSI transport protocol service request to request that a SCSI target port send data over the service delivery subsystem.

Send Data-In SCSI transport protocol service request**:**

> **Send Data-In (IN (I_T_L Nexus, Command Identifier, Device Server Buffer,
> Application Client Buffer Offset, Request Byte Count))**

Input arguments:

I_T_L Nexus:	An identifier for the I_T_L nexus routing the command (see 4.6.32.3).
Command Identifier:	The numerical identifier identifying the command (see 4.6.31.3).
Device Server Buffer:	The buffer in the device server from which data is to be transferred.
Application Client Buffer Offset:	Offset in bytes from the beginning of the application client's buffer (i.e., the Data-In Buffer) to the first byte of transferred data.
Request Byte Count:	Number of bytes to be moved by this request.

5.4.3.2.2 Data-In Delivered SCSI transport protocol service confirmation

A SCSI target port uses the **Data-In Delivered** SCSI transport protocol service confirmation to notify a device server that it has sent data.

Data-In Delivered SCSI transport protocol service confirmation:

> **Data-In Delivered (IN (I_T_L Nexus, Command Identifier, Delivery Result))**

This SCSI transport protocol service confirmation notifies the device server that the specified data was successfully delivered to the application client buffer, or that a service delivery subsystem error occurred while attempting to deliver the data.

Input arguments:

I_T_L Nexus:	An identifier for the I_T_L nexus routing the command (see 4.6.32.3).
Command Identifier:	The numerical identifier identifying the command (see 4.6.31.3).
Delivery Result:	An encoded value representing one of the following: DELIVERY SUCCESSFUL: The data was delivered successfully. DELIVERY FAILURE: A service delivery subsystem error occurred while attempting to deliver the data.

5.4.3.3 Data-Out delivery service

5.4.3.3.1 Receive Data-Out SCSI transport protocol service request

A device server invokes the **Receive Data-Out** SCSI transport protocol service request to request that a SCSI target port receive data over the service delivery subsystem.

Receive Data-Out SCSI transport protocol service request:

> **Receive Data-Out (IN (I_T_L Nexus, Command Identifier, Application Client Buffer Offset,
> Request Byte Count, Device Server Buffer))**

Input arguments:

I_T_L Nexus:	An identifier for the I_T_L nexus routing the command (see 4.6.32.3).
Command Identifier:	The numerical identifier identifying the command (see 4.6.31.3).
Application Client Buffer Offset:	Offset in bytes from the beginning of the application client's buffer (i.e., the Data-Out Buffer) to the first byte of transferred data.
Request Byte Count:	Number of bytes to be moved by this request.
Device Server Buffer:	The buffer in the device server to which data is to be transferred.

If the **SCSI Command Received** SCSI transport protocol service included a First Burst Enabled argument and random buffer access is not supported, first burst data shall be transferred to the Device Server Buffer until all first burst data has been transferred. If the **SCSI Command Received** SCSI transport protocol service included a First Burst Enabled argument and random buffer access is supported, first burst data should be transferred to the Device Server Buffer but first burst data may be re-transferred across a service delivery subsystem.

5.4.3.3.2 Data-Out Received SCSI transport protocol service confirmation

A SCSI target port invokes the **Data-Out Received** SCSI transport protocol service confirmation to notify a device server that it has received data over the service delivery subsystem.

Data-Out Received SCSI transport protocol service confirmation:

Data-Out Received (IN (I_T_L Nexus, Command Identifier, Delivery Result))

This SCSI transport protocol service confirmation notifies the device server that the requested data has been successfully delivered to its buffer, or that a service delivery subsystem error occurred while attempting to receive the data.

Input arguments:

I_T_L Nexus:	An identifier for the I_T_L nexus routing the command (see 4.6.32.3).
Command Identifier:	The numerical identifier identifying the command (see 4.6.31.3).
Delivery Result:	An encoded value representing one of the following: DELIVERY SUCCESSFUL: The data was delivered successfully. DELIVERY FAILURE: A service delivery subsystem error occurred while attempting to receive the data.

5.4.3.4 Terminate Data Transfer service

5.4.3.4.1 Terminate Data Transfer SCSI transport protocol service request

A device server or task manager invokes the **Terminate Data Transfer** SCSI transport protocol service request to request that a SCSI target port terminate data transfers.

Terminate Data Transfer SCSI transport protocol service request:

Terminate Data Transfer (IN (Nexus, [Command identifier]))

Input argument:

Nexus:	An identifier for the I_T nexus (see 4.6.32.2), or an identifier for the I_T_L nexus (see 4.6.32.3).
Command Identifier:	The numerical identifier (see 4.6.31.3) identifying the command associated with the data transfer being terminated.

The SCSI target port terminates all transfer service requests for the specified command identifier on the specified nexus. If no command identifier is specified, then the SCSI target port terminates all transfer service requests for all commands on the specified nexus (e.g., if an I_T_L nexus is specified with no command identifier, then the SCSI target port terminates all transfer service requests from the logical unit for the specified SCSI initiator port).

5.4.3.4.2 Data Transfer Terminated SCSI transport protocol service confirmation

A SCSI target port invokes the **Data Transfer Terminated** SCSI transport protocol service confirmation to notify a device server or task manager that it has terminated all outstanding data transfers for the specified command on the specified nexus.

Data Transfer Terminated SCSI transport protocol service confirmation:

<div align="center">

Data Transfer Terminated (IN (Nexus, [Command identifier]))

</div>

Input argument:

> **Nexus:** An identifier for the I_T nexus (see 4.6.32.2), or an identifier for the I_T_L nexus (see 4.6.32.3).
>
> **Command Identifier:** The numerical identifier (see 4.6.31.3) identifying the command associated with the data transfer being terminated.

This SCSI transport protocol service confirmation is returned in response to a **Terminate Data Transfer** SCSI transport protocol service request whether or not the specified nexus existed in the SCSI target port when the request was received. After a **Data Transfer Terminated** SCSI transport protocol service confirmation has been sent in response to a **Terminate Data Transfer** SCSI transport protocol service request, **Data-In Delivered** SCSI transport protocol service confirmation and **Data-Out Received** SCSI transport protocol service confirmations shall not be sent for the commands specified by the nexus.

5.5 Command lifetime

This subclause specifies the events delimiting the beginning and end (i.e., lifetime) of a command from the viewpoint of the application client and device server. The task router and task manager have the same viewpoint of the beginning and end of a command as the device server.

An application client maintains an application client command (see 4.6.31) from the time the **Send SCSI Command** SCSI transport protocol service request is invoked until:

a) the application client receives a service response of COMMAND COMPLETE for that command;
b) the application client receives notification of a unit attention condition with one of the following additional sense codes:
  A) any additional sense code whose ADDITIONAL SENSE CODE field is set to 2Fh (e.g., COMMANDS CLEARED BY ANOTHER INITIATOR, COMMANDS CLEARED BY POWER LOSS NOTIFICATION or COMMANDS CLEARED BY DEVICE SERVER), if in reference to the task set containing the command;
  B) any additional sense code whose ADDITIONAL SENSE CODE field is set to 29h (e.g., POWER ON, RESET, OR BUS DEVICE RESET OCCURRED; POWER ON OCCURRED; SCSI BUS RESET OCCURRED; BUS DEVICE RESET FUNCTION OCCURRED; DEVICE INTERNAL RESET; or I_T NEXUS LOSS OCCURRED); or
  C) MICROCODE HAS BEEN CHANGED;
c) the application client receives notification that the task manager has detected an overlapped command (see 5.10);
d) the SCSI initiator device processes a SCSI device condition that aborts the command (see table 46);
e) the application client receives notification that the SCSI target device processed a SCSI device condition that aborted the command (see table 47);
f) a service response of FUNCTION COMPLETE for a task management function that aborted the command (see table 48);

g) a service response of COMMAND COMPLETE for another command that aborted the command (see table 49);

h) the application client receives a service response of FUNCTION COMPLETE following a QUERY TASK task management function (see 7.8) directed to the specified command; or

i) the application client receives a service response of FUNCTION COMPLETE following a QUERY TASK SET task management function (see 7.9) directed to the specified task set.

> NOTE 6 - Items other than a) assume in-order delivery (see 4.4.3).

If a service response of SERVICE DELIVERY OR TARGET FAILURE is received for a command (e.g., when an I_T nexus loss is detected by the SCSI initiator port), the application client shall maintain an application client command (see 4.6.31) to represent the command until the application client has determined that the command is no longer known to the device server.

> NOTE 7 - The names of the unit attention conditions listed in the subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in SAM-2. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

The task manager creates a command upon receiving a **SCSI Command Received** SCSI transport protocol service indication (see 5.4.2.3) (i.e., upon processing the SCSI Command Received operation (see 4.6.21.2)).

The command shall exist until:

a) the device server sends a service response for the command of COMMAND COMPLETE; or

b) the command is aborted as described in 5.6.

If a SCSI transport protocol does not require state synchronization (see 4.4.2), then there may be a time skew between the completion of a device server request-response transaction as seen by the application client and device server. As a result, the lifetime of a command as it appears to the application client is different from the lifetime observed by the device server.

Some commands initiate background operations that are processed after the command is no longer in the task set (i.e., status has been returned for the command) (e.g., a SEND DIAGNOSTIC command if used to initiate a background self-test (see SPC-4) or a write command if write cache is enabled (see SBC-3)). Background operations may be aborted by power on, hard reset, or logical unit reset. Unless otherwise specified (see 6.3.4 and 6.3.5), background operations shall not be aborted by I_T nexus loss or power loss expected.

Background operations may generate deferred errors that are reported in the sense data for a subsequent command (see SPC-4). Information that a deferred error occurred may be cleared before it is reported (e.g., by power on, hard reset, or logical unit reset). Deferred errors should not be cleared by I_T nexus loss or power loss expected.

Unless a command completes with GOOD status or CONDITION MET status, the degree to which the required command processing has been completed is vendor specific.

5.6 Aborting commands

A command is aborted if a SCSI device condition (see 6.3), command, or task management function causes termination of the command prior to its completion by the device server. After a command is aborted and TASK ABORTED status, if any, is returned, the SCSI target device shall send no further requests or responses for that command.

See table 46 for a list of the SCSI device conditions that cause commands to be aborted in a SCSI initiator device.

Table 46 — SCSI device conditions that abort commands in a SCSI initiator device

SCSI device condition	Scope	Reference
Power on	All commands in the SCSI initiator device.	6.3.1
Hard reset	All commands with an I_T nexus involving the SCSI initiator port.	6.3.2
I_T nexus loss	All commands associated with the lost I_T nexus.	6.3.4
SCSI transport protocol specific conditions	As defined by the applicable SCSI transport protocol standard.	

See table 47 for a list of the SCSI device conditions that cause commands to be aborted in a SCSI target device.

Table 47 — SCSI device conditions that abort commands in a SCSI target device

SCSI device condition	Scope	Unit attention condition (see 5.14) additional sense code, if any	TASK ABORTED status	Reference
Power on	All commands in all logical units in the SCSI target device.	see table 54	no	6.3.1
Hard reset	All commands in all logical units to which the SCSI target port has access in the SCSI target device.		yes or no [b]	6.3.2
Logical unit reset [a]	All commands in the logical unit.		yes or no [c]	6.3.3 and 7.7
I_T nexus loss [a]	In each logical unit to which the SCSI target port has access, all commands associated with the lost I_T nexus.		no	6.3.4 and 7.6
Power loss expected	All commands in all logical units in the SCSI target device.	COMMANDS CLEARED BY POWER LOSS NOTIFICATION	no	6.3.5
SCSI transport protocol specific conditions	As defined by the applicable SCSI transport protocol standard.			

Key:
yes = Each command that is aborted on an I_T nexus other than the one that caused the SCSI device
     condition is completed with TASK ABORTED status, if the TAS bit is set to one in the Control mode
     page (see SPC-4).
no = No status is returned for aborted commands.

[a] This SCSI device condition is able to be invoked by a task management function listed in table 48.
[b] If the hard reset is caused by a particular I_T nexus (e.g., by a SCSI transport protocol-specific task
    management function), then "yes" applies, otherwise, "no" applies.
[c] If the logical unit reset is caused by a particular I_T nexus (e.g., by a LOGICAL UNIT RESET task
    management function), then "yes" applies, otherwise (e.g., if triggered by a hard reset), "no" applies.

See table 48 for a list of the task management functions that cause commands to be aborted.

Table 48 — Task management functions that abort commands (part 1 of 2)

Task management function	Scope	Unit attention condition (see 5.14) additional sense code, if any [a]	TASK ABORTED status	Reference
ABORT TASK if the ABORT TASK scope is I_T_L nexus	All commands: a) in the specified logical unit; b) in the task set associated with commands from the specified I_T nexus; and c) that were received on the specified I_T nexus with the specified command identifier. [b]	none	no	7.2
ABORT TASK if the ABORT TASK scope is I_T nexus	All commands: a) in all logical units; b) in the task set associated with commands from the specified I_T nexus; and c) that were received on the specified I_T nexus with the specified command identifier. [b]	none	no	7.2

Key:

yes = Each command that is aborted on an I_T nexus other than the one that delivered the task management function is completed with TASK ABORTED status, if the TAS bit is set to one in the Control mode page.

no = No status is returned for aborted commands.

[a] If the TAS bit is set to zero in the Control mode page (see SPC-4), then the device server creates this unit attention condition for each I_T nexus that had command(s) aborted other than the I_T nexus that delivered the task management function. If the TAS bit is set to one in the Control mode page (see SPC-4), then the device server does not create this unit attention condition.

[b] If:
   a) the SCSI target device does not check overlapped commands;
   b) the SCSI transport protocol standard defines that the command identifier scope is the I_T nexus; and
   c) a command in another logical unit is using the same command identifier,
   then the ABORT TASK should not affect commands in other logical units in the same SCSI target device.

[c] If the TST field is set to 001b (i.e., per I_T nexus) in the Control mode page (see SPC-4), then there is one task set per I_T nexus in the logical unit. As a result, commands from other I_T nexuses are not affected and CLEAR TASK SET is equivalent to ABORT TASK SET.

Table 48 — Task management functions that abort commands (part 2 of 2)

Task management function	Scope	Unit attention condition (see 5.14) additional sense code, if any [a]	TASK ABORTED status	Reference
ABORT TASK SET (I_T_L nexus)	All commands: a) in the specified logical unit; b) in the task set associated with commands from the specified I_T nexus; and c) that were received on the specified I_T nexus with any command identifier.	none	no	7.3
CLEAR TASK SET (I_T_L nexus)	All commands: a) in the specified logical unit; b) in the task set associated with commands from the specified I_T nexus; and c) that were received on any I_T nexus with any command identifier. [c]	COMMANDS CLEARED BY ANOTHER INITIATOR	yes	7.5
LOGICAL UNIT RESET (I_T_L nexus)	See table 47 description of the logical unit reset condition.			
I_T NEXUS RESET (I_T nexus)	See table 47 description of the I_T nexus loss condition.			

Key:
yes = Each command that is aborted on an I_T nexus other than the one that delivered the task management function is completed with TASK ABORTED status, if the TAS bit is set to one in the Control mode page.
no = No status is returned for aborted commands.

[a] If the TAS bit is set to zero in the Control mode page (see SPC-4), then the device server creates this unit attention condition for each I_T nexus that had command(s) aborted other than the I_T nexus that delivered the task management function. If the TAS bit is set to one in the Control mode page (see SPC-4), then the device server does not create this unit attention condition.

[b] If:
a) the SCSI target device does not check overlapped commands;
b) the SCSI transport protocol standard defines that the command identifier scope is the I_T nexus; and
c) a command in another logical unit is using the same command identifier,
then the ABORT TASK should not affect commands in other logical units in the same SCSI target device.

[c] If the TST field is set to 001b (i.e., per I_T nexus) in the Control mode page (see SPC-4), then there is one task set per I_T nexus in the logical unit. As a result, commands from other I_T nexuses are not affected and CLEAR TASK SET is equivalent to ABORT TASK SET.

See table 49 for a list of the command related conditions that cause commands to be aborted.

Table 49 — Command related conditions that abort commands

Command related conditions	Scope	Unit attention condition (see 5.14) additional sense code, if any [a]	TASK ABORTED status	Reference
Completion of a command with CHECK CONDITION status if: a) the QERR field is set to 01b; and b) the TST field is set to 000b (i.e., shared) in the Control mode page (see SPC-4).	All commands in the task set.	COMMANDS CLEARED BY ANOTHER INITIATOR	yes	5.8.3 and 5.9.2
Completion of a command with CHECK CONDITION status if: a) the QERR field is set to 01b; and b) the TST field is set to 001b (i.e., per I_T nexus) in the Control mode page (see SPC-4).	All commands in the task set. [b]	none	no	5.8.3 and 5.9.2
Completion of a command with CHECK CONDITION status if the QERR field is set to 11b in the Control mode page (see SPC-4).	All commands in the task set from the same I_T nexus as the command that was terminated.	none	no	5.8.3 and 5.9.2
Processing of a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with a reservation key that is associated with the I_T nexus on which the command was received (see SPC-4).	All commands in the task set from all I_T nexuses with the specified reservation key.	COMMANDS CLEARED BY ANOTHER INITIATOR	yes	SPC-4
Processing of a COPY OPERATION ABORT command that specifies a copy operation that is being processed in the foreground.	The copy operation originated by the third-party copy command specified by the LIST IDENTIFIER field in a COPY OPERATION ABORT command.	none	no	SPC-4
The return of an service response of SERVICE DELIVERY OR TARGET FAILURE.	The indicated command.	none	n/a	5.1
Detection of an overlapped command.	See 5.10	none	no	5.10

Key:
yes = Each command that is aborted on an I_T nexus other than the one that delivered the command is completed with TASK ABORTED status, if the TAS bit is set to one in the Control mode page (see SPC-4).
no = TASK ABORTED status is not returned for aborted commands although other status values may be returned.

[a] If the TAS bit is set to zero in the Control mode page (see SPC-4), then the device server creates this unit attention condition for each I_T nexus that had command(s) aborted other than the I_T nexus that delivered the command. If the TAS bit is set to one in the Control mode page (see SPC-4), then the device server does not create this unit attention condition.
[b] As a result of the TST field being set to 001b, there is one task set per I_T nexus, so no other I_T nexuses are affected.

If one or more commands are cleared or aborted, then the application client removes the affected application client commands (see 5.5). The affected commands are also cleared from the SCSI initiator ports in a manner that is outside the scope of this standard.

If a device server receives a command or task management function on an I_T nexus that causes commands on the same I_T nexus to be aborted, then the device server shall not return any notification that commands have been aborted other than:

 a) a completion response for the command or task management function that caused the command(s) to be aborted; and
 b) notification(s) associated with related effects of the command or task management function (e.g., a reset unit attention condition).

If a device server receives a command or task management function on an I_T nexus that causes commands on other I_T nexuses to be aborted, then in addition to any notification (e.g., completion response) that a command or task management function generates on the I_T nexus on which command or task management function was received, the device server shall return any notifications for those commands on other I_T nexuses based on the setting of the TAS bit in the Control mode page (see SPC-4):

 a) if the TAS bit is set to zero, then the device server:
    A) shall not return status for the commands on other I_T nexuses that were aborted; and
    B) shall establish a unit attention condition for the SCSI initiator port associated with each I_T nexus containing commands on other I_T nexuses that were aborted with an additional sense code set as defined in table 48 and table 49;

    or

 b) if the TAS bit is set to one, then the device server:
    A) shall complete each aborted command on other I_T nexuses with a TASK ABORTED status; and
    B) shall not establish a unit attention condition for this reason.

If a logical unit completes one or more commands received on an I_T nexus with TASK ABORTED status, then the logical unit should complete all of the affected commands before entering any other commands received on that I_T nexus into the task set.

5.7 Command processing example

A command is used to show the events associated with the processing of a single device service request (see figure 39). This example does not include error or exception conditions.
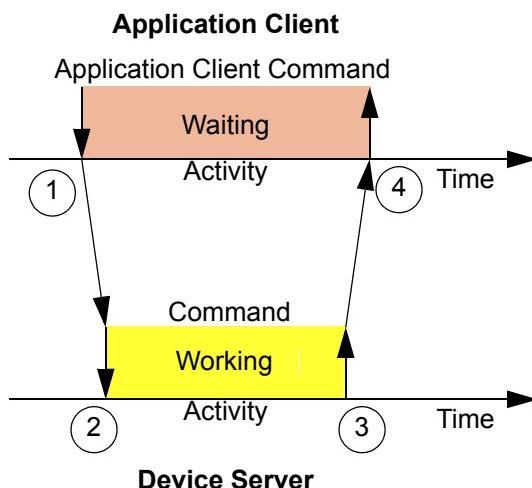
Figure 39 — Command processing events

The numbers in figure 39 identify the events described as follows:

1) the application client performs an **Execute Command** procedure call by invoking the **Send SCSI Command** SCSI transport protocol service request to send the CDB and other input parameters to the logical unit;

2) the task manager is notified through a **SCSI Command Received** SCSI transport protocol service indication containing the CDB and command parameters. A command is created and entered into the task set. The device server may invoke the appropriate data delivery service one or more times to complete command processing;

3) on command completion, the **Send Command Complete** SCSI transport protocol service response is invoked to return GOOD status and a service response of COMMAND COMPLETE; and

4) a **Command Complete Received** SCSI transport protocol service confirmation is passed to the application client by the SCSI initiator port.

5.8 Commands that complete with CHECK CONDITION status

5.8.1 Overview

An application client uses the NACA bit in the CONTROL byte of the CDB (see 5.2) to specify whether or not the device server establishes an ACA condition when it terminates a command with CHECK CONDITION status. The meaning of the value in the NACA bit is as follows:

a) If the NACA bit is set to zero, an ACA condition shall not be established; or
b) If the NACA bit is set to one, an ACA condition shall be established (see 5.9).

The requirements that apply when the ACA condition is not in effect are described in 5.8.2.

If a command terminates with a CHECK CONDITION status and an ACA condition is not established, then commands other than the command completing with the CHECK CONDITION status may be aborted as described in 5.8.3.

5.8.2 Handling commands when ACA is not in effect

Table 50 describes the handling of commands when an ACA condition is not in effect for the task set. The I_T nexuses that are associated with a task set are specified by the TST field in the Control mode page (see SPC-4).

Table 50 — Command handling when ACA is not in effect

New command properties		Device server action	ACA established if new command terminates with a CHECK CONDITION status
Task attribute [a]	**NACA value** [b]		
Any task attribute except the ACA task attribute	0	Process the command. [c]	no
	1		yes
ACA task attribute	0	Process an invalid task attribute condition as described in 5.12.	no
	1		yes
[a] Task attributes are described in 8.4.			
[b] The NACA bit is in the CONTROL byte in the CDB (see 5.2).			
[c] All the conditions that affect the processing of commands (e.g., reservations) apply.			

5.8.3 Aborting commands terminated with a CHECK CONDITION status without establishing an ACA

If a command terminates with a CHECK CONDITION status where the NACA bit is set to zero in the command's CDB CONTROL byte (i.e., when an ACA condition is not established), then commands may be aborted based on the contents of the TST field and QERR field in the Control mode page (see SPC-4 and see 8.7.2.5.1).

5.9 Auto contingent allegiance (ACA)

5.9.1 ACA overview

The application client may request that the device server alter command processing when a command terminates with a CHECK CONDITION status by establishing an ACA condition using the NACA bit in the CONTROL byte (see 5.8.1).

The steps taken by the device server to establish an ACA condition are described in 5.9.2. Upon establishment of the ACA condition, some commands other than the command completing with the CHECK CONDITION status may be aborted and continued processing of other commands may be blocked as described in 5.9.2.

While the ACA condition is in effect and the TMF_ONLY bit is set to zero in the Control mode page (see SPC-4), new commands received by the logical unit from the faulted I_T nexus are not allowed to enter the task set (see 5.9.3) unless they have the ACA task attribute (see 8.4.5). One of the results of the ACA task attribute requirement is that commands in-flight when the CHECK CONDITION status occurs are completed unprocessed with an ACA ACTIVE status. Multiple commands may be sent one at a time (see 8.7.2.5.2) using the ACA task attribute to recover from the event that resulted in the ACA condition without clearing the ACA condition.

While the ACA condition is in effect and the TMF_ONLY bit is set to one, no new commands received by the logical unit from the faulted I_T nexus are allowed to enter the task set.

While the ACA condition is in effect:

   a) new commands received on the faulted I_T nexus shall be handled as described in 5.9.3;
   b) new commands received on I_T nexuses other than the faulted I_T nexus shall be handled as described in 5.9.4; and
   c) commands that were in the task set when the ACA condition occurs shall be handled as described in 8.7.

The methods for clearing an ACA condition are described in 5.9.5.

5.9.2 Establishing an ACA

If a device server terminates a command with a CHECK CONDITION status and the NACA bit was set to one in the CONTROL byte of the terminated command, then the device server shall create an ACA condition.

If an ACA condition is established, then commands shall either be aborted or blocked based on the contents of the TST field and QERR field in the Control mode page (see SPC-4 and see 8.7.2.5.2).

An ACA condition shall not cross task set boundaries and shall be preserved until it is cleared as described in 5.9.5.

If the SCSI transport protocol does not enforce state synchronization as described in 4.4.2, then there may be a time delay between the occurrence of the ACA condition and the time at which the application client becomes aware of the condition.

5.9.3 Handling new commands received on the faulted I_T nexus when ACA is in effect

Table 51 describes the handling of new commands received on the faulted I_T nexus when ACA is in effect.

Table 51 — Handling for new commands received on a faulted I_T nexus during ACA

New command properties		ACA command present in the task set [c]	TMF_ONLY value [d]	Device server action	ACA established if new command terminates with a CHECK CONDITION status
Task attribute [a]	NACA value [b]				
ACA task attribute	0	no	0	Process the command. [f]	no [e]
	1	no	0		yes [e]
	n/a	n/a	1	Complete the command with ACA ACTIVE status.	n/a
	0 or 1	yes	n/a		n/a
Any task attribute except the ACA task attribute	0 or 1	n/a	n/a	Complete the command with ACA ACTIVE status.	n/a

> [a] Task attributes are described in 8.4.
> [b] The NACA bit is in the CONTROL byte in the CDB (see 5.2).
> [c] A command with the ACA task attribute (see 8.4.5).
> [d] The TMF_ONLY bit is in the Control mode page (see SPC-4).
> [e] If a command with the ACA task attribute terminates with a CHECK CONDITION status, the existing ACA condition shall be cleared and the value of the NACA bit shall control the establishment of a new ACA condition.
> [f] All the conditions that affect the processing of commands (e.g., reservations) apply.

5.9.4 Handling new commands received on non-faulted I_T nexuses when ACA is in effect

5.9.4.1 Command processing that is permitted for commands received on a non-faulted I_T nexuses during ACA

The device server shall process a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action (see SPC-4) while an ACA condition is established if the command is received on a non-faulted I_T nexus.

> NOTE 8 - The processing of specific commands (e.g., PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action) received on a non-faulted I_T nexus while an ACA condition is in effect provides SCSI initiator ports not associated with the faulted I_T nexus the opportunity to recover from error conditions that the SCSI initiator port associated with the faulted I_T nexus is unable to recover from itself.

5.9.4.2 Handling new commands received on non-faulted I_T nexuses when ACA is in effect

The handling of commands received on I_T nexuses other than the faulted I_T nexus depends on the value in the TST field in the Control mode page (see SPC-4).

Table 52 describes the handling of new commands received on I_T nexuses other than the faulted I_T nexus when ACA is in effect.

Table 52 — Handling for new commands received on non-faulted I_T nexuses during ACA

TST field value in Control mode page	New command properties		New command permitted during ACA [c]	Device server action	ACA established if new command terminates with a CHECK CONDITION status
	Task attribute [a]	NACA value [b]			
000b	ACA task attribute	n/a	n/a	Complete the command with ACA ACTIVE status.	n/a
	Any task attribute except the ACA task attribute	0	no	Complete the command with BUSY status.	n/a
		1	no	Complete the command with ACA ACTIVE status.	n/a
		0	yes	Process the command.	no [d]
		1	yes		yes [d]
001b	ACA task attribute	0	n/a	Process an invalid task attribute condition as described in 5.12.	no
		1			yes
	Any task attribute except the ACA task attribute	0 or 1	n/a	Process the command. [e]	See 5.8.2.

[a] Task attributes are described in 8.4.
[b] The NACA bit is in the CONTROL byte in the CDB (see 5.2).
[c] See 5.9.4.1.
[d] If a permitted command terminates with CHECK CONDITION status, the existing ACA condition shall be cleared and the value of the NACA bit shall control the establishment of a new ACA condition.
[e] If the TST field in the Control mode page contains 001b, then commands received on a non-faulted I_T nexus shall be processed as if the ACA condition does not exist (see 5.8.2). In this case, the logical unit shall be capable of handling concurrent ACA conditions and sense data associated with each I_T nexus.

5.9.5 Clearing an ACA condition

An ACA condition shall only be cleared:

a) as a result of a hard reset (see 6.3.2), logical unit reset (see 6.3.3), or I_T nexus loss (see 6.3.4);
b) by a CLEAR ACA task management function (see 7.4) received on the faulted I_T nexus;
c) by a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with the ACA task attribute received on the faulted I_T nexus that clears the commands received on the faulted I_T nexus (see SPC-4);
d) by a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with a task attribute other than the ACA task attribute received on a non-faulted I_T nexus that clears the commands received on the faulted I_T nexus;

e) if a command with the ACA task attribute received on the faulted I_T nexus terminates with CHECK
CONDITION status; or

f) if a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action
terminates with CHECK CONDITION status.

Cases e) and f) may result in the establishment of a new ACA based on the value of the NACA bit.

If an ACA condition is cleared and no new ACA condition is established, then the state of all commands in the
task set shall be modified as described in 8.7.

When the ACA condition is cleared, any command remaining in that task set may be completed subject to the
requirements for task set management described in clause 8.

5.10 Overlapped commands

An overlapped command occurs when a task manager or a task router detects the use of a duplicate
command identifier (see 4.6.31.3) in a command before that command completes its command lifetime (see
5.5). Each SCSI transport protocol standard shall specify whether or not a task manager or a task router is
required to detect overlapped commands.

If the SCSI transport protocol defines that the scope of the command identifier is the I_T nexus, the SCSI
target device aborts overlapped commands, and the task router detects an overlapped command, then the
task router:

a) shall abort all commands (see 5.6) in all logical units that were received on the I_T nexus on which
the overlapped command was received;

b) shall abort all task management functions in all logical units that were received on the I_T nexus on
which the overlapped command was received if required by the SCSI transport protocol standard;

c) should abort all task management functions in all logical units that were received on the I_T nexus on
which the overlapped command was received if not required by the SCSI transport protocol standard;
and

d) shall invoke Send Command Complete operation (see 4.6.6.7) of the SCSI target port specified by
the I_T nexus to return a CHECK CONDITION status for the overlapped command with the sense key
set to ABORTED COMMAND and the additional sense code set to OVERLAPPED COMMANDS
ATTEMPTED.

If the SCSI transport protocol defines that the scope of the command identifier is the I_T nexus, the SCSI
target device aborts overlapped commands, and a task manager detects an overlapped command, then that
task manager:

a) shall abort all commands (see 5.6) in that logical unit that were received on the I_T nexus on which
the overlapped command was received;

b) shall abort all task management functions in that logical unit that were received on the I_T nexus on
which the overlapped command was received if required by the SCSI transport protocol standard;

c) should abort all task management functions in that logical unit that were received on the I_T nexus on
which the overlapped command was received if not required by the SCSI transport protocol standard;
and

d) shall invoke Send Command Complete operation (see 4.6.6.7) of the SCSI target port specified by
the I_T nexus to return a CHECK CONDITION status for the overlapped command with the sense key
set to ABORTED COMMAND and the additional sense code set to OVERLAPPED COMMANDS
ATTEMPTED.

If the SCSI transport protocol defines that the scope of the command identifier is the I_T_L nexus, the SCSI
target device aborts overlapped commands, and the task router or a task manager detects an overlapped
command, then that task router or task manager:

a) shall abort all commands (see 5.6) in that logical unit that were received on the I_T nexus on which
the overlapped command was received;

b) shall abort all task management functions in that logical unit that were received on the I_T nexus on
which the overlapped command was received if required by the SCSI transport protocol standard;

c) should abort all task management functions in that logical unit that were received on the I_T nexus on which the overlapped command was received if not required by the SCSI transport protocol standard; and

d) shall invoke Send Command Complete operation (see 4.6.6.7) of the SCSI target port specified by the I_T nexus to return a CHECK CONDITION status for the overlapped command with the sense key set to ABORTED COMMAND and the additional sense code set to OVERLAPPED COMMANDS ATTEMPTED.

NOTE 9 - An overlapped command may be indicative of a serious error and, if not detected, may result in corrupted data. This is considered a catastrophic failure on the part of the SCSI initiator device. Therefore, vendor specific error recovery procedures may be required to guarantee the data integrity on the medium. The additional sense data may aid in this error recovery procedure (e.g., sequential-access devices may terminate the overlapped command with the residue of blocks remaining to be written or read at the time the second command was received).

5.11 Incorrect logical unit numbers for commands

The SCSI target device's processing of an incorrect logical unit number (see 4.7.1 and 4.6.25.2.2) is described in this subclause.

In response to a REQUEST SENSE command, a REPORT LUNS command, or an INQUIRY command the SCSI target device shall respond as defined in SPC-4.

If a command other than a REQUEST SENSE command, a REPORT LUNS command, or an INQUIRY command is not rerouted to an administrative logical unit (see 4.6.7.2), then that command:

a) shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and with the additional sense code set to LOGICAL UNIT NOT SUPPORTED, if:
   A) the SCSI target device is not capable of supporting the logical unit (e.g., some SCSI target devices support only one logical unit); or
   B) the SCSI target device supports the logical unit, but the peripheral device is not currently connected to the SCSI target device;

   or

b) is responded to in a vendor specific manner, if:
   A) the SCSI target device supports the logical unit and that logical unit is accessible to the task router contained in the SCSI target port that received the command, however, that logical unit is not operational; or
   B) the SCSI target device supports the logical unit but is incapable of determining if that logical unit is accessible, at this time, to the task router contained in the SCSI target port that received the command.

If a command other than a REQUEST SENSE command, a REPORT LUNS command, or an INQUIRY command to a subsidiary logical unit is rerouted to an administrative logical unit (see 4.6.7.2) for processing, then that command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and with the additional sense code set to SUBSIDIARY LOGICAL UNIT NOT CONFIGURED as described in 4.6.25.2.2.

5.12 Task attribute exception conditions

If a command is received with a task attribute that is not supported or is not valid (e.g., an ACA task attribute if an ACA condition does not exist), then the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID MESSAGE ERROR.

> NOTE 10 - The use of the INVALID MESSAGE ERROR additional sense code is based on its similar usage in
> SAM-2. The use of the INVALID MESSAGE ERROR additional sense code is not to be interpreted as a
> description of how the task attributes are represented by any given SCSI transport protocol.

Task attribute support should be reported with the Extended INQUIRY Data VPD page (see SPC-4).

5.13 Sense data

5.13.1 Command terminated sense data or polled sense data

Sense data shall be made available by the logical unit if a command terminates with CHECK CONDITION
status or other conditions (e.g., the processing of a REQUEST SENSE command). The format, content, and
conditions under which sense data shall be prepared by the logical unit are as defined in this standard,
SPC-4, the applicable command standard, and the applicable SCSI transport protocol standard.

Sense data associated with an I_T nexus shall be preserved by the logical unit until:

    a) the sense data is transferred;
    b) a logical unit reset (see 6.3.3) occurs;
    c) an I_T nexus loss (see 6.3.4) occurs for the I_T nexus associated with the preserved sense data; or
    d) power loss expected (see 6.3.5) occurs.

If a command terminates with CHECK CONDITION status, then sense data shall be returned in the same I_T
nexus transaction as the CHECK CONDITION status. After the sense data is returned, it shall be cleared
except if it is associated with a unit attention condition and the UA_INTLCK_CTRL field in the Control mode page
(see SPC-4) contains 10b or 11b.

Completion with sense data in the same I_T nexus transaction as a CHECK CONDITION status shall not
affect ACA (see 5.9) or the sense data associated with a unit attention condition if the UA_INTLCK_CTRL field
contains 10b or 11b.

5.13.2 Command completed sense data

Sense data may be made available by the logical unit if a command completes with a status other than
CHECK CONDITION (e.g., GOOD). The format, content, and conditions under which sense data shall be
prepared by the logical unit are as defined in this standard, SPC-4, the applicable command standard, and the
applicable SCSI transport protocol standard.

If a command completes with a status other than CHECK CONDITION (e.g., GOOD), the D_SENSE bit in the
Control Mode page (see SPC-4) is set to one (i.e., returning descriptor formatted sense data is enabled), and
sense data is available, then that sense data shall be returned in the same I_T nexus transaction as the
status. After the sense data is returned, the sense data may be cleared.

5.14 Unit attention conditions

5.14.1 Unit attention conditions that are not coalesced

Each logical unit shall establish a unit attention condition whenever one of the following events occurs:

    a) a power on (see 6.3.1), hard reset (see 6.3.2), logical unit reset (see 6.3.3), I_T nexus loss (see
        6.3.4), or power loss expected (see 6.3.5) occurs;
    b) commands received on this I_T nexus have been cleared by a command or a task management
        function associated with another I_T nexus and the TAS bit was set to zero in the Control mode page
        associated with this I_T nexus (see 5.6);
    c) the portion of the logical unit inventory that consists of administrative logical units and hierarchical
        logical units has been changed (see 4.6.18.1); or
    d) any other event requiring the attention of the SCSI initiator device.

Each logical unit shall establish a unit attention condition when an I_T nexus is established that has not been previously established. The additional sense code shall be set to:

a) POWER ON, RESET, OR BUS DEVICE RESET OCCURRED;
b) DEVICE INTERNAL RESET; or
c) POWER ON OCCURRED.

The conditions that cause a unit attention condition to be established may no longer exist at the time the unit attention is received by the application client.

Unit attention conditions are classified by precedence levels. Table 53 defines the unit attention condition precedence levels.

Table 53 — Unit attention condition precedence level

Unit attention condition additional sense code	Unit attention condition precedence
POWER ON, RESET, OR BUS DEVICE RESET OCCURRED	highest
POWER ON OCCURRED or DEVICE INTERNAL RESET	
SCSI BUS RESET OCCURRED or MICROCODE HAS BEEN CHANGED or protocol specific	
BUS DEVICE RESET FUNCTION OCCURRED	
I_T NEXUS LOSS OCCURRED	
COMMANDS CLEARED BY POWER LOSS NOTIFICATION	
all others	Lowest

For unit attention conditions with the lowest precedence level with a given ADDITIONAL SENSE CODE field value, the unit attention condition with the ADDITIONAL SENSE CODE QUALIFIER field set to 00h has higher precedence level than the unit attention conditions with the ADDITIONAL SENSE CODE QUALIFIER field set to values other than 00h (e.g., PARAMETERS CHANGED has precedence over MODE PARAMETERS CHANGED and LOG PARAMETERS CHANGED). A unit attention condition with the lowest precedence level has equal priority with all unit attention conditions with the lowest precedence level with different ADDITIONAL SENSE CODE field values.

> NOTE 11 - The unit attention additional sense code specificity order defined in 6.2 determines which unit attention condition is allowed to be established when certain conditions occur. The unit attention condition precedence defined in this subclause determines which unit attention conditions are allowed to clear other unit attention conditions if they have not yet been reported.

The device server shall maintain a queue of unit attention conditions of undefined order for each I_T nexus. The queue should be large enough to hold every unit attention condition that the device server is capable of reporting.

When a device server establishes a unit attention condition:

1) the device server may clear unit attention conditions from the queue that are no longer needed as follows:
    A) the device server may clear any pending unit attention conditions in the queue that have lower precedence levels (e.g., BUS DEVICE RESET FUNCTION OCCURRED may clear I_T NEXUS LOSS OCCURRED and all unit attention conditions with a lower precedence); and
    B) the device server should clear pending unit attention conditions that have the same additional sense code (i.e., the device server should not add the same unit attention condition twice);
2) if a queue slot is available, then:

    A) if a higher precedence unit attention condition is not in the queue, then the device server shall add the unit attention condition to the queue; or

    B) if a higher precedence unit attention condition is in the queue, then the device server should add the unit attention condition to the queue,

in the sense data for the unit attention condition, the device shall either:

    A) not include sense key specific sense data; or

    B) include sense key specific sense data and set the OVERFLOW bit to zero (see SPC-4);

or

3) if a queue slot is not available, then the device server shall either:
    A) replace any unit attention condition in the queue; or
    B) not add the unit attention condition to the queue,

the device server shall include sense key specific sense data and set the OVERFLOW bit to one (see SPC-4) for at least one unit attention condition in the queue.

If the device server establishes multiple unit attention conditions as a result of the same event or a series of events, then it may establish the unit attention conditions in any order (e.g., in direct-access block devices, if a MODE SELECT command changes the initial command priority value, then the device server may report PRIORITY CHANGED before MODE PARAMETERS CHANGED or may report MODE PARAMETERS CHANGED before PRIORITY CHANGED).

When the device server reports and clears a unit attention condition, it:

    a) may select any unit attention condition in the queue to report; and
    b) shall clear the unit attention condition from the queue after reporting it.

A unit attention condition shall persist until the device server clears the unit attention condition. Unit attention conditions are affected by the processing of commands as follows:

    a) if an INQUIRY command enters the enabled command state, then the device server shall process the INQUIRY command and shall neither report nor clear any unit attention condition;
    b) if a REPORT LUNS command enters the enabled command state, then the device server shall process the REPORT LUNS command and shall not report any unit attention condition.

If the UA_INTLCK_CTRL field in the Control mode page is set to 00b (see SPC-4) and the logical unit processing the command:

    A) is a subsidiary logical unit, then the SCSI target device shall clear any pending unit attention condition with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the SCSI initiator port associated with that I_T nexus in each logical unit associated with the logical unit conglomerate that contains the subsidiary logical unit processing the REPORT LUNS command that is accessible by that I_T nexus. Other pending unit attention conditions shall not be cleared; or
    B) is not a subsidiary logical unit, then the SCSI target device shall clear any pending unit attention condition with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the SCSI initiator port associated with that I_T nexus in each logical unit accessible by the I_T nexus on which the REPORT LUNS command was received. Other pending unit attention conditions shall not be cleared.

If the UA_INTLCK_CTRL field in the Control mode page contains 10b or 11b, the SCSI target device shall not clear any unit attention condition(s);

    c) if a REQUEST SENSE command enters the enabled command state while a unit attention condition exists for the SCSI initiator port associated with the I_T nexus on which the REQUEST SENSE command was received, then the device server shall process the command and either:
    A) report any pending sense data as parameter data and preserve all unit attention conditions on the logical unit; or
    B) report a unit attention condition as parameter data for the REQUEST SENSE command to the SCSI initiator port associated with the I_T nexus on which the REQUEST SENSE command was received. The logical unit may discard any pending sense data and shall clear the reported unit

attention condition for the SCSI initiator port associated with that I_T nexus. If the unit attention condition has an additional sense code of REPORTED LUNS DATA HAS CHANGED and the logical unit processing the command:

    a) is a subsidiary logical unit, then the SCSI target device shall clear any pending unit attention condition with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the I_T nexus in each logical unit associated with the logical unit conglomerate that contains the subsidiary logical unit processing the REQUEST SENSE command that is accessible by that I_T nexus; or

    b) is not a subsidiary logical unit, then the SCSI target device shall clear any pending unit attention conditions with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the I_T nexus on which the command was received in each logical unit accessible by that I_T nexus;

If the device server has already generated the ACA condition (see 5.9) for a unit attention condition, then the device server shall report the unit attention condition (i.e., option B) of c) above);

    d) if the device server supports the NOTIFY DATA TRANSFER DEVICE command (see ADC-2) and a NOTIFY DATA TRANSFER DEVICE command enters the enabled command state, then the device server shall process the NOTIFY DATA TRANSFER DEVICE command and shall neither report nor clear any unit attention condition; and

    e) if a command other than INQUIRY, REPORT LUNS, REQUEST SENSE, or NOTIFY DATA TRANSFER DEVICE enters the enabled command state while a unit attention condition exists for the SCSI initiator port associated with the I_T nexus on which the command was received, then the device server shall terminate the command with a CHECK CONDITION status. The device server shall provide sense data that reports a unit attention condition for the SCSI initiator port that sent the command on the I_T nexus.

If a device server reports a unit attention condition with a CHECK CONDITION status and the UA_INTLCK_CTRL field in the Control mode page contains 00b (see SPC-4), then the device server shall clear the reported unit attention condition for the SCSI initiator port associated with that I_T nexus on the logical unit. If the unit attention condition has an additional sense code of REPORTED LUNS DATA HAS CHANGED and the logical unit on which the unit attention condition was reported:

    A) is a subsidiary logical unit, then the SCSI target device shall clear any pending unit attention conditions with the additional sense code of REPORTED LUNS DATA HAS CHANGED established for the SCSI initiator port associated with the I_T nexus on each logical unit associated with the logical unit conglomerate that contains the subsidiary logical unit that is accessible by that I_T nexus; or

    B) is not a subsidiary logical unit, then the SCSI target device shall clear any pending unit attention conditions with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the I_T nexus on which the command was received in each logical unit accessible by that I_T nexus.

If the UA_INTLCK_CTRL field contains 10b or 11b, then the device server shall not clear unit attention conditions reported with a CHECK CONDITION status.

5.14.2 Coalescing unit attention conditions

Within a logical unit conglomerate (see 4.6.10), the unit attention conditions associated with the following additional sense codes that occur in a subsidiary logical unit (see 4.6.26) shall be coalesced by the administrative logical unit (see 4.6.11) as described in this subclause:

    a) REPORTED LUNS DATA HAS CHANGED if that unit attention condition is a result of a change in the inventory of subsidiary logical units;

    b) ASYMMETRIC ACCESS STATE CHANGED; and

    c) IMPLICIT ASYMMETRIC ACCESS STATE TRANSITION FAILED.

If a device server in a subsidiary logical unit is required to establish a unit attention that is reported using one of the additional sense codes listed in this subclause, then the device server shall:

    a) not establish the unit attention condition; and

    b)   indicate to the administrative logical unit for the logical unit conglomerate that the administrative logical unit's device server is required to establish the unit attention condition.

If an administrative logical unit's device server is notified that it is required to establish a unit attention condition on behalf of one or more of the subsidiary logical units in the logical unit conglomerate, then the administrative logical unit's device server shall:

1) establish a unit attention condition for the additional sense code specified by the subsidiary logical unit or units and maintain it in a coalesced unit attentions queue that has all the properties of the queue described in 5.14.1 with the additional requirement that coalesced unit attention conditions have a separate identifying characteristic;

2) invoke the Reroute Conglomerate Task Management Functions operation (see 4.6.7.5) to reroute all task management functions to the administrative logical unit's task manager for processing as described in 4.6.25.2.4;

3) use the algorithm described in 4.6.11 to reroute commands associated with subsidiary logical units in the logical unit conglomerate in a way that causes the CHECK CONDITION status for the unit attention condition to be returned for a command sent to:

    A)   the administrative logical unit; or

    B)   any of the subsidiary logical units;

    and

4) after the unit attention condition has been returned, invoke the Stop Conglomerate Task Management Functions Rerouting operation (see 4.6.7.6) to end rerouting of task management functions within the logical unit conglomerate.

6 SCSI events and event notification model

6.1 SCSI events overview

SCSI events may occur or be detected in either:

   a) the SCSI device;
   b) one or more SCSI ports within a SCSI device; or
   c) the application client, task manager, or device server.

The detection of any event may require processing by the object that detects it.

Events that occur in a SCSI device are assumed to be detected and processed by all objects within the SCSI device.

When a SCSI port detects an event, it shall use the event notification services (see 6.4) to notify task managers or application clients that the event has been detected.

The events detected and event notification services usage depends on whether the SCSI device is a SCSI target device (see figure 40) or a SCSI initiator device (see figure 41).
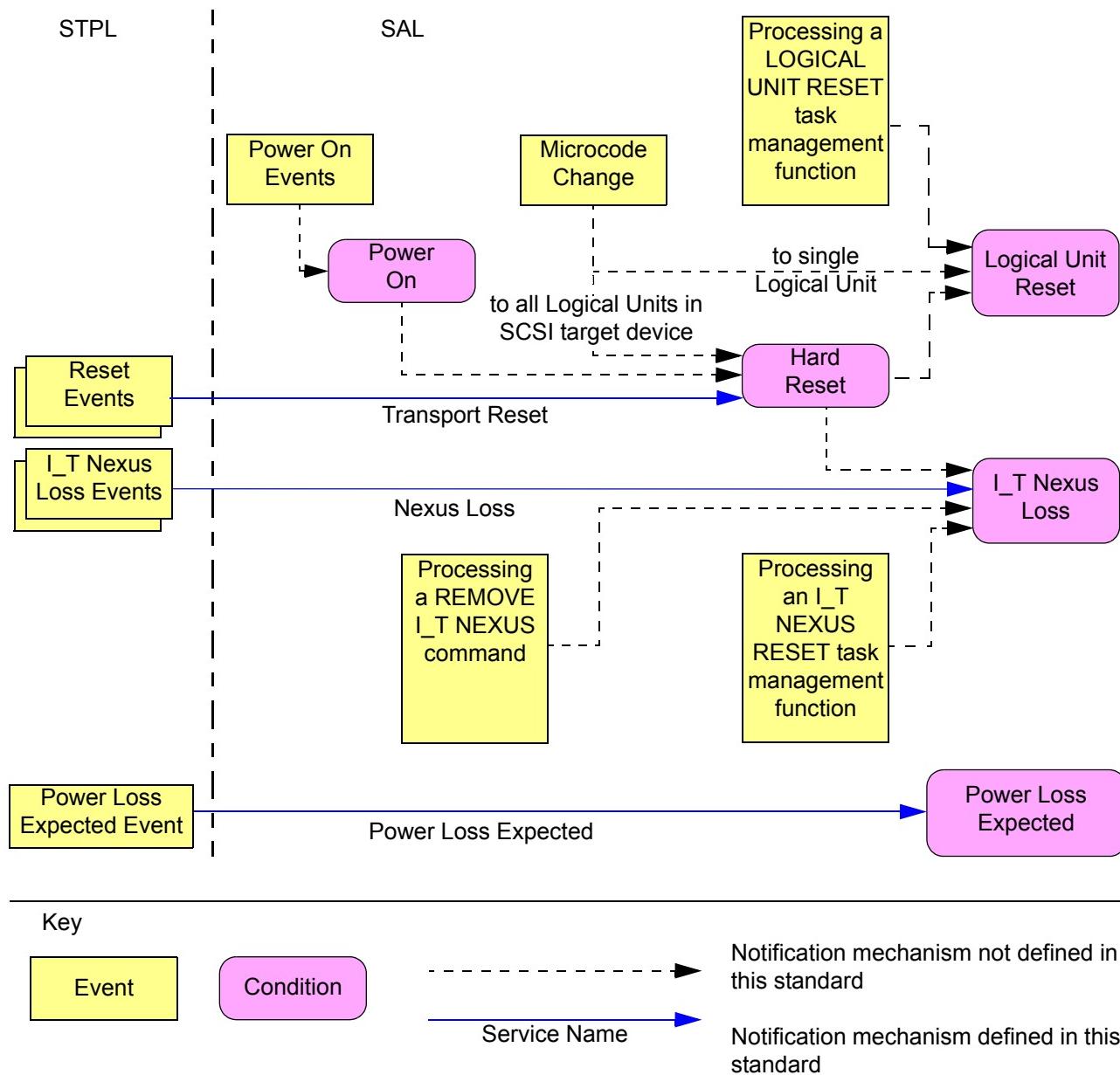
Figure 40 — Events and event notifications for SCSI target devices
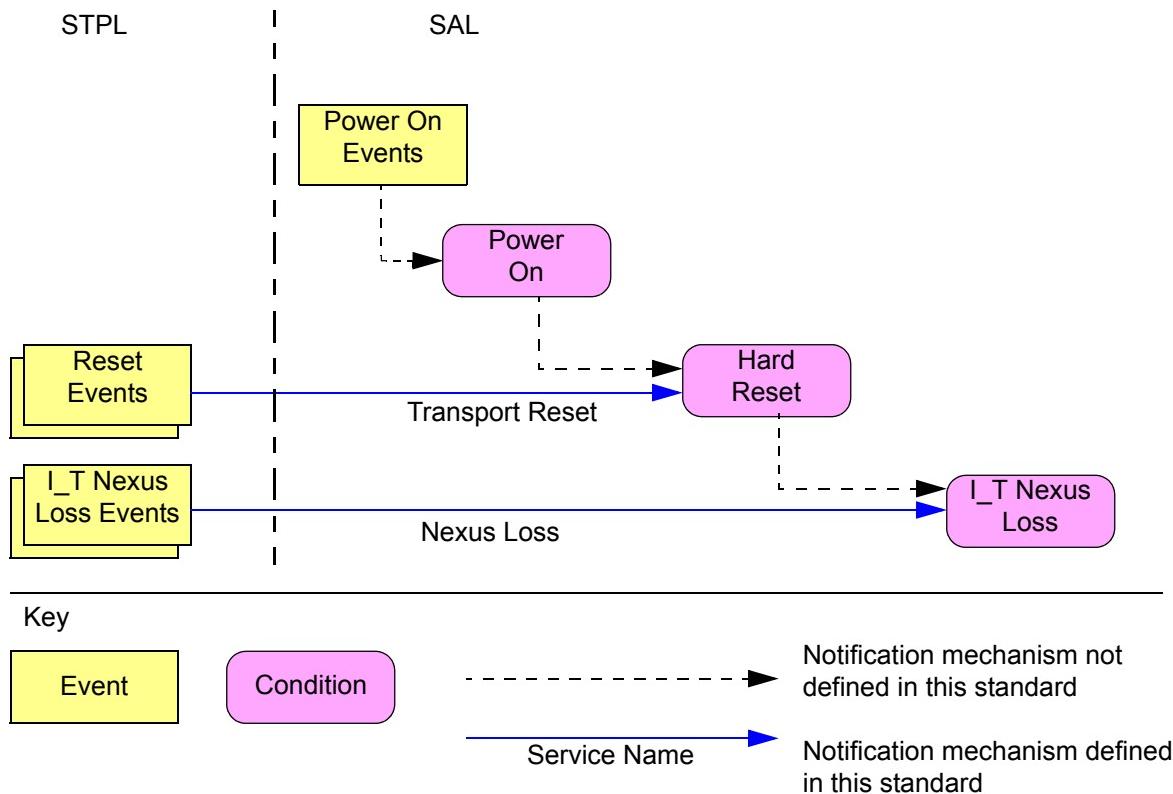
Figure 41 — Events and event notifications for SCSI initiator devices

6.2 Establishing a unit attention condition subsequent to detection of an event

Table 54 shows the additional sense code that a logical unit shall use when a unit attention condition (see 5.14) is established for each of the conditions shown in figure 40 (see 6.1). A SCSI transport protocol may define additional sense codes that are more specific than SCSI BUS RESET OCCURRED for reset events. The most specific condition known to the logical unit should be used to establish the additional sense code for a unit attention.

The unit attention additional sense code specificity order defined in this subclause determines which unit attention condition is allowed to be established when certain conditions occur. The unit attention condition precedence defined in 5.14 determines which unit attention conditions are allowed to clear other unit attention conditions if they have not yet been reported.

Table 54 — Unit attention additional sense codes for events detected by SCSI target devices

Condition	Additional sense code	Specificity
Logical unit is unable to distinguish between the conditions	POWER ON, RESET, OR BUS DEVICE RESET OCCURRED	Lowest
Power on	POWER ON OCCURRED or DEVICE INTERNAL RESET [a]	
Hard reset	SCSI BUS RESET OCCURRED or MICROCODE HAS BEEN CHANGED [b] or protocol specific [c]	
Logical unit reset	BUS DEVICE RESET FUNCTION OCCURRED or MICROCODE HAS BEEN CHANGED [b]	
I_T nexus loss	I_T NEXUS LOSS OCCURRED	
Power loss expected	COMMANDS CLEARED BY POWER LOSS NOTIFICATION	Highest
[a] Used after a vendor-specific power on event has occurred (e.g., a firmware reboot). [b] Only used if microcode has been changed (see SPC-4). [c] Only used if a protocol-specific reset event has occurred.		

NOTE 12 - The names of the unit attention conditions listed in this subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in SAM-2. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

A logical unit should use the I_T NEXUS LOSS OCCURRED additional sense code when establishing a unit attention condition for an I_T nexus loss if:

    a)   the SCSI initiator port to which the sense data is being delivered is the SCSI initiator port that was associated with the I_T nexus loss, and the logical unit has maintained all state information specific to that SCSI initiator port since the I_T nexus loss; or

    b)   the I_T nexus being used to deliver the sense data is the same I_T nexus that was lost, and the logical unit has maintained all state information specific to that I_T nexus since the I_T nexus loss,

otherwise, the logical unit shall use one of the less specific additional sense codes (e.g., POWER ON OCCURRED) when establishing a unit attention condition for an I_T nexus loss.

6.3 Conditions resulting from SCSI events

6.3.1 Power on

Power on is a SCSI device condition resulting from a power on event. When a SCSI device is powered on, it shall cause a hard reset.

The power on condition applies to both SCSI initiator devices and SCSI target devices.

Power on events include:

   a) power being applied to the SCSI device; and
   b) vendor-specific events that cause the SCSI device to behave as if power has been applied (e.g.,
       firmware reboot).

6.3.2 Hard reset

Hard reset is a SCSI device condition resulting from:

   a) a power on condition (see 6.3.1);
   b) microcode change (see SPC-4); or
   c) a reset event indicated by a **Transport Reset** event notification (see 6.4).

The definition of reset events and the notification of their detection is SCSI transport protocol specific.

Each SCSI transport protocol standard that defines reset events shall specify a SCSI target port's protocol
specific actions in response to reset events. Each SCSI transport protocol standard that defines reset events
should specify when those events result in the delivery of a **Transport Reset** event notification to the SCSI
application layer.

SCSI transport protocols may include reset events that have no effects on SCSI devices attached to the SCSI
transport (e.g., a Fibre Channel non-initializing loop initialization primitive).

The hard reset condition applies to both SCSI initiator devices and SCSI target devices.

A SCSI target port's response to a hard reset condition shall include a logical unit reset condition (see 6.3.3)
for all logical units to which the SCSI target port has access. A hard reset condition shall not affect any other
SCSI target ports in the SCSI target device, however, the logical unit reset condition established by a hard
reset may affect commands and task management functions that are communicating via other SCSI target
ports.

Although the task manager response to task management requests is subject to the presence of access
restrictions, as managed by ACCESS CONTROL OUT commands (see SPC-4), a hard reset condition shall
not be prevented by access controls.

If a SCSI initiator port detects a hard reset condition, then it should terminate all its outstanding **Execute
Command** procedure calls and all its outstanding task management function procedure calls with a service
response of SERVICE DELIVERY OR TARGET FAILURE. A hard reset condition shall not affect any other SCSI
initiator ports in the SCSI initiator device, however, the logical unit reset condition established in a SCSI target
device by a hard reset may affect commands and task management functions that are communicating via
other SCSI initiator ports.

A SCSI port's response to a hard reset condition shall include establishing an I_T nexus loss condition (see
6.3.4) for every I_T nexus associated with that SCSI port.

6.3.3 Logical unit reset

Logical unit reset is a SCSI device condition for a logical unit resulting from:

   a) a hard reset condition (see 6.3.2);
   b) a microcode change; or
   c) a logical unit reset event indicating that a LOGICAL UNIT RESET task management request (see 7.7)
       has been processed.

The logical unit reset condition applies only to SCSI target devices.

When responding to a logical unit reset condition, the logical unit shall:

   a) abort all commands as described in 5.6;
   b) abort all copy operations (see SPC-4);
   c) terminate all task management functions;

    d)   clear all ACA conditions (see 5.9.5) in all task sets in the logical unit;
    e)   establish a unit attention condition (see 5.14 and 6.2);
    f)   initiate a logical unit reset for all dependent logical units (see 4.6.18.4);
    g)   stop rerouting of task management functions (see 4.6.7.5);
    h)   discard task rerouting information (see 4.6.7.4); and
    i)   perform any additional functions required by the applicable command standards.

6.3.4 I_T nexus loss

An I_T nexus loss is a SCSI device condition resulting from:

    a)   a hard reset condition (see 6.3.2);
    b)   an I_T nexus loss event (e.g., logout) indicated by a **Nexus Loss** event notification (see 6.4);
    c)   indication that an I_T NEXUS RESET task management request (see 7.6) has been processed; or
    d)   an indication that a REMOVE I_T NEXUS command (see SPC-4) has been processed.

An I_T nexus loss event is an indication from the SCSI transport protocol to the SAL that an I_T nexus no longer exists. SCSI transport protocols may define I_T nexus loss events.

Each SCSI transport protocol standard that defines I_T nexus loss events should specify when those events result in the delivery of a **Nexus Loss** event notification to the SAL.

The I_T nexus loss condition applies to both SCSI initiator devices and SCSI target devices.

If a SCSI target port detects an I_T nexus loss, then a **Nexus Loss** event notification shall be delivered to each logical unit to which the I_T nexus has access.

In response to an I_T nexus loss condition a logical unit shall take the following actions:

    a)   abort all commands received on the I_T nexus as described in 5.6;
    b)   abort all background third-party copy operations (see SPC-4) that are using the I_T nexus;
    c)   terminate all task management functions received on the I_T nexus;
    d)   clear all ACA conditions (see 5.9.5) associated with the I_T nexus;
    e)   establish a unit attention condition for the SCSI initiator port associated with the I_T nexus (see 5.14 and 6.2); and
    f)   perform any additional functions required by the applicable command standards.

If the logical unit retains state information for the I_T nexus that is lost, its response to the subsequent I_T nexus re-establishment for the logical unit should include establishing a unit attention with an additional sense code set to I_T NEXUS LOSS OCCURRED.

If the logical unit does not retain state information for the I_T nexus that is lost, it shall consider the subsequent I_T nexus re-establishment, if any, as the formation of a new I_T nexus for which there is no past history (e.g., establish a unit attention with an additional sense code set to POWER ON OCCURRED).

If a SCSI initiator port detects an I_T nexus loss, then it should terminate all its outstanding **Execute Command** procedure calls and all its outstanding task management function procedure calls for the SCSI target port associated with the I_T nexus with a service response of SERVICE DELIVERY OR TARGET FAILURE.

6.3.5 Power loss expected

Power loss expected is a SCSI device condition resulting from a power loss expected event indicated by a Power Loss Expected event notification (see 6.4).

A power loss expected event is an indication from the SCSI transport protocol to the SAL that power loss may occur within a protocol specific period of time. SCSI transport protocols may define power loss expected events.

Each SCSI transport protocol standard that defines power loss expected events should specify when those events result in the delivery of a Power Loss Expected event notification to the SAL.

If a SCSI target port detects a power loss expected, then a Power Loss Expected event notification indication shall be delivered to each logical unit to which the I_T nexus has access. In response to the resulting I_T power loss expected condition:

a) the task manager in each logical unit to which the notification was delivered shall:
  A) abort all commands in all task sets (see 5.6), clearing all pending status and sense data;
  B) abort all background third-party copy operations (see SPC-4);
  C) establish a unit attention condition as described in 5.6 for each initiator port associated with every I_T nexus on which a command was aborted (e.g., with the additional sense code set to COMMANDS CLEARED BY POWER LOSS NOTIFICATION); and
  D) abort all task management functions;
b) each logical unit to which the notification was delivered shall not change any other previously established conditions, including mode parameters, reservations, and ACA; and
c) the SCSI target device shall perform any additional functions required by the applicable SCSI transport protocol standards.

6.4 SCSI transport protocol services for event notification

6.4.1 SCSI transport protocol service for event notification overview

The SCSI transport protocol services described in this subclause are used by a SCSI initiator port or a SCSI target port to deliver an indication to the SAL that a SCSI event has been detected.

All SCSI transport protocol standards should define the SCSI transport protocol specific requirements for implementing the **Nexus Loss** SCSI transport protocol service indication, the **Transport Reset** SCSI transport protocol service indication and the **Power Loss Expected** SCSI transport protocol service indication described in this subclause and when these indications are to be delivered to the SAL.

The **Nexus Loss** SCSI transport protocol service indication and the **Transport Reset** SCSI transport protocol service indication are defined for both SCSI target devices and SCSI initiator devices.

6.4.2 Nexus Loss SCSI transport protocol service indication

A SCSI target port invokes the **Nexus Loss** SCSI transport protocol service indication to notify a task manager that an I_T nexus loss has occurred.

A SCSI initiator port invokes the **Nexus Loss** SCSI transport protocol service indication to notify an application client that an I_T nexus loss has occurred.

<div align="center">**Nexus Loss (IN (I_T Nexus))**</div>

Input argument

      I_T Nexus: An identifier for the specific I_T nexus that has been detected as lost.

6.4.3 Transport Reset SCSI transport protocol service indication

A SCSI target port invokes the **Transport Reset** SCSI transport protocol service indication to notify a task manager that a hard reset has occurred.

A SCSI initiator port invokes the **Transport Reset** SCSI transport protocol service indication to notify an application client that a hard reset has occurred.

<div align="center">**Transport Reset   (IN (SCSI Port))**</div>

Input argument

      SCSI Port: The specific SCSI port in the SCSI device for which a transport reset was detected.

6.4.4 Power Loss Expected SCSI transport protocol service indication

The **Power Loss Expected** SCSI transport protocol service indication is defined for SCSI target devices.

A SCSI target port invokes the **Power Loss Expected** SCSI transport protocol service indication to notify a task manager that a power loss expected has occurred.

Power Loss Expected   (IN (SCSI Port))

Input argument

> **SCSI Port:** The specific SCSI port in the SCSI device for which power loss expected was detected.

7 Task management functions

7.1 Task management function procedure calls

An application client requests the processing of a task management function by invoking the SCSI transport protocol services described in 7.12, the collective operation of which is modeled in the following procedure call using the following format:

Service Response = Function name   (IN (Nexus, Command Identifier), OUT ([Additional Response Information])

The task management function names are summarized in table 55.

Table 55 — Task Management Functions

Task management function (i.e., function name)	Nexus argument	Command Identifier argument required	Additional Response Information argument supported	Reference
ABORT TASK	I_T_L Nexus	yes	no	7.2
ABORT TASK SET	I_T_L Nexus	no	no	7.3
CLEAR ACA	I_T_L Nexus	no	no	7.4
CLEAR TASK SET	I_T_L Nexus	no	no	7.5
I_T NEXUS RESET	I_T Nexus	no	no	7.6
LOGICAL UNIT RESET	I_T_L Nexus	no	no	7.7
QUERY TASK	I_T_L Nexus	yes	yes	7.8
QUERY TASK SET	I_T_L Nexus	no	no	7.9
QUERY ASYNCHRONOUS EVENT	I_T_L Nexus	no	yes	7.10

Input arguments:

Nexus: Contains an I_T Nexus argument (see 4.6.32.2), or an I_T_L Nexus argument (see 4.6.32.3) identifying the routing of the command or commands affected by the task management function.

I_T Nexus: An identifier for the I_T nexus (see 4.6.32.2) affected by the task management function.

I_T_L Nexus: An identifier for the I_T_L nexus (see 4.6.32.3) affected by the task management function.

Command Identifier: The numerical identifier (see 4.6.31.3) affected by the task management function.

Output arguments:

Additional Response Information: If supported by the SCSI transport protocol and the logical unit, then three bytes are returned along with the service response for certain task management functions (e.g., QUERY ASYNCHRONOUS EVENT). SCSI transport protocols may or may not support the Additional Response Information argument. A SCSI transport protocol supporting the Additional Response Information argument may or may not require that logical units accessible through a SCSI target port using that transport protocol support the Additional Response Information argument. If the SCSI transport protocol does not support Additional Response Information or the logical unit does not support Additional Response Information, then all output parameters are invalid.

One of the following SCSI transport protocol specific service responses shall be returned:

FUNCTION COMPLETE: A task manager response indicating that the requested function is complete. Unless another response is required, the task manager shall return this response upon completion of a task management request supported by the logical unit or SCSI target device to which the request was directed.

FUNCTION SUCCEEDED: A task manager response indicating that the requested function is supported and completed successfully. This task manager response shall only be used by functions that require notification of success (e.g., QUERY TASK, QUERY TASK SET, or QUERY ASYNCHRONOUS EVENT).

FUNCTION REJECTED: A task manager response indicating that the requested function is not supported by the logical unit or SCSI target device to which the function was directed.

INCORRECT LOGICAL UNIT NUMBER: A task router response indicating that the function requested processing for an incorrect logical unit number (see 4.7.1).

SERVICE DELIVERY OR TARGET FAILURE: The request was terminated due to a service delivery failure or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

Each SCSI transport protocol standard shall define the events for each of these service responses.

The task manager response to task management requests is subject to the presence of access restrictions, as managed by ACCESS CONTROL OUT and ACCESS CONTROL IN commands (see SPC-4), as follows:

a) a task management request of ABORT TASK, ABORT TASK SET, CLEAR ACA, I_T NEXUS RESET, QUERY TASK, QUERY TASK SET, or QUERY ASYNCHRONOUS EVENT shall not be affected by the presence of access restrictions;

b) a task management request of CLEAR TASK SET or LOGICAL UNIT RESET received from a SCSI initiator port that is denied access to the logical unit, either because it has no access rights or because it is in the pending-enrolled state, shall not cause any changes to the logical unit; and

c) the task management function service response shall not be affected by the presence of access restrictions.

7.2 ABORT TASK

Procedure call:

Service Response =   **ABORT TASK (IN (I_T_L Nexus, Command Identifier))**

Description:

If the SCSI transport protocol for a SCSI target port defines that the scope of the ABORT TASK is:

   a) I_T_L nexus, then the task manager in each logical unit accessible through that SCSI target port shall
       support ABORT TASK; or
   b) I_T nexus, then the task router shall support ABORT TASK.

If the SCSI transport protocol defines that the scope of the ABORT TASK is I_T_L nexus, then the task
manager shall:

   1) abort all commands with the matching command identifier as described in 5.6; and
   2) return a service response of FUNCTION COMPLETE.

If the SCSI transport protocol defines that the scope of the ABORT TASK is I_T nexus, then the task router
shall:

   1) abort all commands that were received on the specified I_T nexus with the matching command
       identifier in all logical units as described in 5.6; and
   2) return a service response of FUNCTION COMPLETE.

If the SCSI target device supports overlapped command checking (see 5.10), then ABORT TASK affects no
more than one command.

The service response of FUNCTION COMPLETE does not distinguish between whether a command was aborted
or no matching command was in the task set.

Previously established conditions, including mode parameters, reservations, and ACA shall not be changed
by the ABORT TASK task management function.

All SCSI transport protocol standards shall support the ABORT TASK task management function.

7.3 ABORT TASK SET

Procedure call:

<p align="center">Service Response =   ABORT TASK SET (IN (I_T_L Nexus))</p>

Description:

This function shall be supported by all logical units.

The task manager shall:

   1) abort all commands in the task set that were received on the specified I_T nexus as described in 5.6;
       and
   2) return a service response of FUNCTION COMPLETE.

Commands received on other I_T nexuses or in other task sets shall not be aborted. This task management
function performed is equivalent to a series of ABORT TASK requests.

All pending status and sense data for the commands that were aborted shall be cleared. Other previously
established conditions, including mode parameters, reservations, and ACA shall not be changed by the
ABORT TASK SET function.

All SCSI transport protocol standards shall support the ABORT TASK SET task management function.

7.4 CLEAR ACA

Procedure call:

<p align="center">Service Response = CLEAR ACA (IN (I_T_L Nexus))</p>

Description:

This function shall be supported by a logical unit if it supports ACA (see 5.9).

For the CLEAR ACA task management function, the task set shall be the one defined by the TST field in the Control mode page (see SPC-4).

An application client requests a CLEAR ACA using the faulted I_T nexus to clear an ACA condition (see 5.9) from the task set serviced by the logical unit. If there is an ACA condition and this function is received on the faulted I_T nexus, then the task manager shall:

1) abort each command in the task set with the ACA task attribute (see 8.4.5) as if an ABORT TASK function (see 7.2) specifying that command had occurred;
2) clear the ACA condition as described in 5.9.5; and
3) return a service response of FUNCTION COMPLETE.

If there is an ACA condition and this function is received on an I_T nexus other than the faulted I_T nexus, then the task manager shall return a service response of FUNCTION REJECTED.

If there is no ACA condition, then the task manager shall return a service response of FUNCTION COMPLETE.

All SCSI transport protocol standards shall support the CLEAR ACA task management function.

7.5 CLEAR TASK SET

Procedure call:

Service Response = CLEAR TASK SET (IN (I_T_L Nexus))

Description:

This function shall be supported by all logical units.

The task manager shall:

1) abort all commands in the task set as described in 5.6; and
2) return a service response of FUNCTION COMPLETE.

If the TST field is set to 001b (i.e., per I_T nexus) in the Control mode page (see SPC-4), there is one task set per I_T nexus. As a result, no other I_T nexuses are affected and CLEAR TASK SET is equivalent to ABORT TASK SET (see 7.2).

All pending status and sense data for the task set shall be cleared. Other previously established conditions, including mode parameters, reservations, and ACA shall not be changed by the CLEAR TASK SET function.

All SCSI transport protocol standards shall support the CLEAR TASK SET task management function.

7.6 I_T NEXUS RESET

Procedure call:

Service Response = I_T NEXUS RESET (IN (I_T Nexus))

Description:

SCSI transport protocols may or may not support I_T NEXUS RESET and may or may not require logical units accessible through SCSI target ports using such transport protocols to support I_T NEXUS RESET.

Each logical unit accessible through the SCSI target port shall perform the I_T nexus loss functions described in 6.3.4 for the I_T nexus on which the function request was received, then:

1) the SCSI target device shall return a service response of FUNCTION COMPLETE; and
2) the logical unit(s) and the SCSI target port shall perform any additional functions specified by the SCSI transport protocol.

7.7 LOGICAL UNIT RESET

Procedure call:

Service Response = LOGICAL UNIT RESET (IN (I_T_L Nexus))

Description:

This function shall be supported by all logical units.

The logical unit shall:

1) perform the logical unit reset functions described in 6.3.3; and
2) return a service response of FUNCTION COMPLETE.

> NOTE 13 - Previous versions of this standard only required LOGICAL UNIT RESET support in logical units that supported hierarchical LUN structures.

All SCSI transport protocol standards shall support the LOGICAL UNIT RESET task management function.

7.8 QUERY TASK

Procedure call:

Service Response = QUERY TASK (IN (I_T_L Nexus, Command Identifier), OUT ([Additional Response Information]))

Description:

SCSI transport protocols may or may not support QUERY TASK and may or may not require SCSI target ports using such transport protocols to support QUERY TASK.

If:

a) the SCSI target port supports QUERY TASK; and
b) the SCSI transport protocol defines that the scope of the QUERY TASK is I_T_L nexus,

then the task manager in each logical unit accessible through that SCSI target port shall support QUERY TASK.

If:

a) the SCSI target port supports QUERY TASK; and
b) the SCSI transport protocol defines that the scope of the QUERY TASK is I_T nexus,

then the task router shall support QUERY TASK.

If the SCSI transport protocol defines that the scope of the QUERY TASK is I_T_L nexus, then the task manager in the specified logical unit shall:

a) if a command with the specified command identifier is present in the task set, then return a service response of FUNCTION SUCCEEDED; or
b) if a command with the specified command identifier is not present in the task set, then return a service response of FUNCTION COMPLETE.

If the SCSI transport protocol defines that the scope of the QUERY TASK is I_T nexus, then the task router shall:

a) if a command with the specified command identifier is present in the task set of any logical unit, then return a service response of FUNCTION SUCCEEDED; or
b) if a command with the specified command identifier is not present in the task set of any logical unit, then return a service response of FUNCTION COMPLETE.

If the service response is not FUNCTION SUCCEEDED, then the task router or task manager shall set the Additional Response Information argument to 000000h.

If the service response is FUNCTION SUCCEEDED, then the task router or task manager shall set the Additional Response Information argument as defined in table 56.

Table 56 — Additional Response Information argument for QUERY TASK

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				PROGRESS INDICATION				(LSB)
2	Reserved							

The PROGRESS INDICATION field is a percent complete indication of the specified command. The returned value is a numerator that has 65 536 (i.e., 10000h) as its denominator. The progress indication shall be based upon the total operation. A value of one indicates the minimum progress, and a value of FFFFh indicates the maximum progress. A value of zero indicates that the progress indicator is not supported.

The progress indication should be time related. The granularity of these steps should be small enough to provide reasonable assurances to the application client that progress is being made.

7.9 QUERY TASK SET

Procedure call:

Service Response = QUERY TASK SET (IN (I_T_L Nexus))

Description:

SCSI transport protocols may or may not support QUERY TASK SET and may or may not require logical units accessible through SCSI target ports using such transport protocols to support QUERY TASK SET.

The task manager in the specified logical unit shall:

    a) if there is any command present in the task set from the specified I_T nexus, then return a service
        response of FUNCTION SUCCEEDED; or
    b) if there is no command present in the task set from the specified I_T nexus, then return a service
        response of FUNCTION COMPLETE.

7.10 QUERY ASYNCHRONOUS EVENT

Procedure call:

**Service Response = QUERY ASYNCHRONOUS EVENT (IN (I_T_L Nexus), OUT ([Additional
                                                                    Response Information]))**

Description:

A SCSI transport protocol may or may not support QUERY ASYNCHRONOUS EVENT. A SCSI transport protocol supporting QUERY ASYNCHRONOUS EVENT may or may not require logical units accessible through SCSI target ports using that transport protocol to support QUERY ASYNCHRONOUS EVENT.

If a QUERY ASYNCHRONOUS EVENT task management function is rerouted from a subsidiary logical unit to an administrative logical unit (see 4.6.7.5), then the logical unit specified by the QUERY ASYNCHRONOUS EVENT is the subsidiary logical unit (see 4.6.26) to which the task management function was originally directed and that subsidiary logical unit is the specified logical unit.

A coalesced unit attention condition (see 5.14.2) established by an administrative logical unit (see 4.6.25) device server is a pending unit attention condition for that administrative logical unit and all subsidiary logical units (see 4.6.26) in the same logical unit conglomerate (see 4.6.10).

The task manager in the specified logical unit shall:

   a) if there is a unit attention condition (see 5.14) or a deferred error (see SPC-4) pending for the specified I_T nexus, then return a service response of FUNCTION SUCCEEDED; or
   b) if there is no unit attention condition and no deferred error pending for the specified I_T nexus, then return a service response of FUNCTION COMPLETE.

If the service response is not FUNCTION SUCCEEDED, then the task manager shall set the Additional Response Information argument to 000000h.

If the service response is FUNCTION SUCCEEDED, then the task manager shall set the Additional Response Information argument as defined in table 57.

Table 57 — Additional Response Information argument for QUERY ASYNCHRONOUS EVENT

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved		UADE DEPTH		SENSE KEY			
1	ADDITIONAL SENSE CODE							
2	ADDITIONAL SENSE CODE QUALIFIER							

The UADE DEPTH field indicates the number of pending unit attention conditions or deferred errors and is defined in table 58. The number of pending unit attention conditions, shall be based on the content of all applicable unit attention queues (e.g., the administrative logical unit's coalesced unit attentions queue and the subsidiary logical unit's unit attention queue for a rerouted QUERY ASYNCHRONOUS EVENT task management function).

Table 58 — UADE DEPTH field

Code	Description
00b	The combined number of unit attention conditions and deferred errors is unknown.
01b	The combined number of unit attention conditions and deferred errors is one.
10b	The combined number of unit attention conditions and deferred errors is greater than one.
11b	Reserved

The SENSE KEY field indicates the value of the SENSE KEY field that is to be returned in the sense data for the next unit attention condition or deferred error that is to be reported (see SPC-4).

The ADDITIONAL SENSE CODE field indicates the value of the ADDITIONAL SENSE CODE field in the next unit attention condition or deferred error that is to be reported (see SPC-4).

The ADDITIONAL SENSE CODE QUALIFIER field indicates the value of the ADDITIONAL SENSE CODE QUALIFIER field in the next unit attention condition or deferred error that is to be reported (see SPC-4).

The next unit attention condition that is to be reported, if any, shall be based on the highest precedence condition (see 5.3.3) in all applicable unit attention queues (e.g., the administrative logical unit's coalesced unit attentions queue and the subsidiary logical unit's unit attention queue for a rerouted QUERY ASYNCHRONOUS EVENT task management function).

7.11 Task management function lifetime

The task manager creates a task management function upon receiving a **Task Management Request Received** SCSI transport service indication (see 7.12) (i.e., upon processing the Task Management Request Received operation (see 4.6.21.3)). The task management function shall exist until:

    a)   the task manager sends a service response for the task management function;
    b)   an I_T nexus loss (see 6.3.4);
    c)   a logical unit reset (see 6.3.3);
    d)   a hard reset (see 6.3.2);
    e)   power loss expected (see 6.3.5); or
    f)   a power on condition (see 6.3.1).

An application client maintains an application client task management function to represent the task management function from the time the **Send Task Management Request** SCSI transport protocol service request is invoked until the application client receives one of the following SCSI target device responses:

    a)   a service response of FUNCTION COMPLETE, FUNCTION SUCCEEDED, FUNCTION REJECTED, or INCORRECT LOGICAL UNIT NUMBER is received for that task management function;
    b)   notification of a unit attention condition with any additional sense code whose ADDITIONAL SENSE CODE field is set to 29h (e.g., POWER ON, RESET, OR BUS DEVICE RESET OCCURRED; POWER ON OCCURRED; SCSI BUS RESET OCCURRED; BUS DEVICE RESET FUNCTION OCCURRED; DEVICE INTERNAL RESET; or I_T NEXUS LOSS OCCURRED);
    c)   notification of a unit attention condition with an additional sense code set to MICROCODE HAS BEEN CHANGED; or
    d)   notification of a unit attention condition with an additional sense code set to COMMANDS CLEARED BY POWER LOSS NOTIFICATION.

    NOTE 14 - Items other than a) assume in-order delivery (see 4.4.3).

If a service response of SERVICE DELIVERY OR TARGET FAILURE is received for a task management function (e.g., an I_T nexus loss is detected by the SCSI initiator port), the application client shall maintain an application client task management function to represent the task management function until the application client has determined that the task management function is no longer known to the device server.

    NOTE 15 - The names of the unit attention conditions listed in the subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in SAM-2. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

If a SCSI transport protocol does not require state synchronization (see 4.4.2), then there may be a time skew between the completion of a device server request-response transaction as seen by the application client and device server. As a result, the lifetime of a task management function as it appears to the application client is different from the lifetime observed by the device server.

7.12 SCSI transport protocol services for task management functions

7.12.1 SCSI transport protocol services for task management functions overview

The SCSI transport protocol services described in this subclause are used by a SCSI initiator device and SCSI target device to process a task management function procedure call. The following arguments are passed:

> **Nexus:** An identifier for the I_T nexus (see 4.6.32.2), or an identifier for the I_T_L nexus (see 4.6.32.3).
>
> **Command Identifier:** The numerical identifier (see 4.6.31.3) of the command affected by the task management function.
>
> **Function Identifier:** Argument encoding the task management function to be performed.

All SCSI transport protocol standards shall define the SCSI transport protocol specific requirements for implementing the **Send Task Management Request** SCSI transport protocol service request (see 7.12.2), the **Task Management Request Received** SCSI transport protocol service indication (see 7.12.3), the **Task Management Function Executed** SCSI transport protocol service response (see 7.12.4), and the **Received Task Management Function Executed** SCSI transport protocol service confirmation (see 7.12.5) SCSI transport protocol services.

A SCSI transport protocol standard may specify different implementation requirements for the **Send Task Management Request** SCSI transport protocol service request for different values of the Function Identifier argument.

All SCSI initiator devices shall implement the **Send Task Management Request** SCSI transport protocol service request and the **Received Task Management Function Executed** SCSI transport protocol service confirmation as defined in the applicable SCSI transport protocol standards.

All SCSI target devices shall implement the **Task Management Request Received** SCSI transport protocol service indication and the **Task Management Function Executed** SCSI transport protocol service response as defined in the applicable SCSI transport protocol standards.

7.12.2 Send Task Management Request SCSI transport protocol service request

An application client invokes the **Send Task Management Request** SCSI transport protocol service request to request that a SCSI initiator port send a task management function over the service delivery subsystem.

Send Task Management Request SCSI transport protocol service request:

> **Send Task Management Request (IN (Nexus, [Command Identifier], Function Identifier))**

Input arguments:

> **Nexus:** An identifier for the I_T nexus (see 4.6.32.2), or an identifier for the I_T_L nexus (see 4.6.32.3).
>
> **Command Identifier:** The numerical identifier (see 4.6.31.3) of the command affected by the task management function.
>
> **Function Identifier:** Argument encoding the task management function to be performed.

7.12.3 Task Management Request Received SCSI transport protocol service indication

A task router (see 4.6.7) invokes the **Task Management Request Received** SCSI transport protocol service indication to notify a task manager that it has received a task management function over the service delivery subsystem.

Task Management Request Received SCSI transport protocol service indication:

> **Task Management Request Received (IN (Nexus, [Command Identifier], Function Identifier))**

Input arguments:

> **Nexus:** An identifier for the I_T nexus (see 4.6.32.2), or an identifier for the I_T_L nexus (see 4.6.32.3).
>
> **Command Identifier:** The numerical identifier (see 4.6.31.3) of the command affected by the task management function.
>
> **Function Identifier:** Argument encoding the task management function to be performed.

7.12.4 Task Management Function Executed SCSI transport protocol service response

A task manager invokes the **Task Management Function Executed** SCSI transport protocol service response to request that a SCSI target port transmit task management function executed information over the service delivery subsystem.

Task Management Function Executed SCSI transport protocol service response:

Task Management Function Executed (IN (Nexus, [Command Identifier], Service Response, [Additional Response Information]))

Input arguments:

> **Nexus:** An identifier for the I_T nexus (see 4.6.32.2), or an identifier for the I_T_L nexus (see 4.6.32.3).
>
> **Command Identifier:** The numerical identifier (see 4.6.31.3) of the command affected by the task management function.
>
> **Service Response:** An encoded value representing one of the following:
>
>> FUNCTION COMPLETE: The requested function has been completed.
>>
>> FUNCTION SUCCEEDED: The requested function is supported and completed successfully.
>>
>> FUNCTION REJECTED: The task manager does not implement the requested function.
>>
>> INCORRECT LOGICAL UNIT NUMBER: A task router response indicating that the function requested processing for an incorrect logical unit number (see 4.7.1).
>>
>> SERVICE DELIVERY OR TARGET FAILURE: The request was terminated due to a service delivery failure or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.
>
> **Additional Response Information:** The Additional Response Information output argument for the task management function procedure call (see 7.1).

7.12.5 Received Task Management Function Executed SCSI transport protocol service confirmation

A SCSI initiator port invokes the **Received Task Management Function Executed** SCSI transport protocol service confirmation to notify an application client that it has received task management function executed information over the service delivery subsystem.

Received Task Management Function Executed SCSI transport protocol service confirmation:

Received Task Management Function Executed (IN (Nexus, [Command Identifier], Service Response, [Additional Response Information]))

Input arguments:

Nexus:	An identifier for the I_T nexus (see 4.6.32.2), or an identifier for the I_T_L nexus (see 4.6.32.3).
Command Identifier:	The numerical identifier (see 4.6.31.3) of the command affected by the task management function.
Service Response:	An encoded value representing one of the following:

FUNCTION COMPLETE:	The requested function has been completed.
FUNCTION SUCCEEDED:	The requested function is supported and completed successfully.
FUNCTION REJECTED:	The task manager does not implement the requested function.
INCORRECT LOGICAL UNIT NUMBER:	A task router response indicating that the function requested processing for an incorrect logical unit number (see 4.7.1).
SERVICE DELIVERY OR TARGET FAILURE:	The request was terminated due to a service delivery failure or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

Additional Response Information:	The Additional Response Information output argument for the task management function procedure call (see 7.1).

Each SCSI transport protocol shall allow a **Received Task Management Function Executed** confirming completion of the requested task to be associated with the corresponding **Send Task Management Request**.

7.13 Task management function example

Figure 42 shows the sequence of events associated with a task management function.
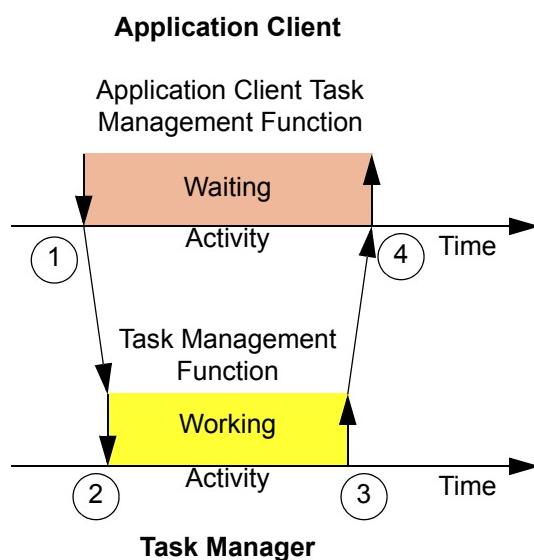
Figure 42 — Task management processing events

The numbers in figure 42 identify the events described as follows:

1) the application client issues a task management request by invoking the **Send Task Management Request** SCSI transport protocol service;
2) the task manager is notified through a **Task Management Request Received** SCSI transport protocol service indication by a task router and begins processing the function;
3) the task manager performs the requested function and responds by invoking the task router's **Task Management Function Executed** SCSI transport protocol service response to notify the application client of a service response of FUNCTION COMPLETE; and
4) a **Received Task Management Function Executed** SCSI transport protocol service confirmation is received by the application client.

8 Task set management

8.1 Task set management overview

This clause describes some of the controls that application clients have over task set management behaviors (see 8.3). This clause also specifies task set management requirements in terms of:

a) LU state machines (see 8.7);
b) task attributes (see 8.4); and
c) command priority (see 8.5).

This clause concludes with several task set management examples (see 8.8).

Command behavior, as described in this clause, refers to the functioning of a command as observed by an application client, including the results of command processing and interactions with other commands.

The requirements for task set management only apply to a command after it has been entered into a task set. A command shall be entered into a task set as defined by the LU_TM state machine (see 8.7.2) unless SCSI transport protocol specific errors cause that command to be terminated with CHECK CONDITION status.

8.2 Implicit head of queue

A command standard may define commands each of which may be processed by the task manager as if the command's task attribute is HEAD OF QUEUE task attribute even if the command is received with a SIMPLE task attribute or an ORDERED task attribute.

An application client should not send a command with the ORDERED task attribute if the command may be processed as if it has a task attribute of HEAD OF QUEUE task attribute because whether the ORDERED task attribute is honored is vendor specific.

8.3 Command management model

The command management model requires the following task set management behaviors:

a) the SIMPLE task attribute (see 8.4.1) shall be supported;
b) task attributes other than SIMPLE may be supported;
c) the QUEUE ALGORITHM MODIFIER field in the Control mode page (see SPC-4) shall control the processing sequence of commands having the SIMPLE task attribute;
d) the QERR field in the Control mode page (see SPC-4) shall control aborting of commands when any command terminates with CHECK CONDITION status; and
e) the CLEAR TASK SET task management function (see 7.5) shall be supported.

8.4 Task attributes

8.4.1 Overview

The application client shall assign a task attribute (see table 59) to each command.

Table 59 — Task attributes

Task attribute	Reference
SIMPLE	8.4.2
ORDERED	8.4.3
HEAD OF QUEUE	8.4.4
ACA	8.4.5

SCSI transport protocols shall provide the capability to specify a unique task attribute for each command.

8.4.2 Commands having the SIMPLE task attribute

If a command having the SIMPLE task attribute is received by the task manager, then the task manager sends that command to the task set (see 8.7.2.2). The task manager does not permit that command to request processing by the device server until all commands having a HEAD OF QUEUE task attribute and older commands having an ORDERED task attribute in the task set have completed (see 8.7.2.2).

8.4.3 Commands having the ORDERED task attribute

If a command having the ORDERED task attribute is received by the task manager, then the task manager sends that command to the task set (see 8.7.2.2). The task manager does not permit that command to request processing by the device server until all commands having a HEAD OF QUEUE task attribute and all older commands in the task set have completed (see 8.7.2.2).

8.4.4 Commands having the HEAD OF QUEUE task attribute

If a command having the HEAD OF QUEUE task attribute is received by the task manager, then the task manager sends that command to the task set with no restrictions on processing by the device server (see 8.7.2.2).

8.4.5 Commands having the ACA task attribute

If a command having the ACA task attribute is received by the task manager, then the task manager sends that command to the task set with no restrictions on processing by the device server (see 8.7.2.2). There shall be no more than one command having the ACA task attribute per task set (see 5.9.2).

8.5 Command priority

Command priority specifies the relative scheduling importance of a command having a SIMPLE task attribute in relation to other commands having SIMPLE task attributes already in the task set. If the command has a task attribute other than SIMPLE, then command priority is not used. Command priority is a value in the range of 0h to Fh. See table 60 for the scheduling importance of the command priority values.

Table 60 — Command priority

Value	Description
0h	A command with either no command priority or a command with a vendor-specific level of scheduling importance.
1h	A command with the highest scheduling importance.
	A command with decreased scheduling importance.
Fh	A command with the lowest scheduling importance.

If the Command Priority argument is set to zero or is not contained within the SCSI Command Received SCSI transport protocol service indication (see 5.4.2.3), and a priority has been assigned to the I_T_L nexus, then the device server shall use the specified priority for the I_T_L nexus as the command priority. A priority is assigned to an I_T_L nexus by a SET PRIORITY command (see SPC-4) or by the INITIAL COMMAND PRIORITY field in the Control Extension mode page (see SPC-4). If no priority has been assigned to the I_T_L nexus using the SET PRIORITY command and the logical unit does not support the INITIAL COMMAND PRIORITY field in the Control Extension mode page, then the device server shall set the command priority to 0h (i.e., vendor specific), or the command shall have no command priority.

A device server may use command priority to determine an ordering to process commands with the SIMPLE task attribute. A difference in command priority between commands may not override other scheduling considerations (e.g., different times to access different logical block addresses) or vendor specific scheduling considerations. However, processing of a collection of commands with different command priorities should cause the subset of commands with the higher command priorities to complete with status sooner in aggregate than if the same collection of commands were submitted under the same conditions with all command priorities being equal.

8.6 Command duration limit

8.6.1 Command duration limit overview

A command duration limit specifies the scheduling and processing of a duration limited command. A duration limited command is a command that:

a) has a SIMPLE task attribute or an ORDERED task attribute;
b) reports a nonzero value in the CDLP field of the REPORT SUPPORTED OPERATION CODES all_commands parameter data or the REPORT SUPPORTED OPERATION CODES one_command parameter data (see SPC-5); and
c) selects a nonzero command duration limit in the Command Duration Limit A mode page or the Command Duration Limit B mode page (see SPC-5).

If the device server has requests to process duration limited commands with a SIMPLE task attribute, then the device server should process the commands based on the command:

1) duration scheduling as defined in 8.6.2; and

2) priority scheduling as defined in 8.5.

If the device server has requests to process duration limited commands with an ORDERED task attribute, then the device server shall process the command as described in 8.7.3.2.

If the device server determines that it is unable to complete a duration limited command before the duration expiration time, then the device server shall terminate the command as described in 8.7.3.2.

The duration expiration time is calculated as follows:

$$\text{duration expiration time} = \text{command arrival time} + \text{command duration limit}$$

where:

>   command arrival time   is the time at which the SCSI Command Received transport protocol service indication (see 5.4.2.3) is invoked; and

>   command duration limit   is as specified in the duration limit descriptor bits in the CDB (see appropriate command standard).

8.6.2 Command duration scheduling

A command duration limit provides information used as part of scheduling of a duration limited command with a SIMPLE task attribute in relation to other commands having SIMPLE task attributes that are already in the task set.

The device server may use the duration expiration time to determine the order of processing commands with the SIMPLE task attribute within the task set. A difference in duration expiration time between commands may override other scheduling considerations (e.g., different times to access different logical block addresses or vendor specific scheduling considerations). Processing of a collection of commands with different command duration limit settings should cause a command with an earlier duration expiration time to complete with status sooner than a command with a later duration expiration time.

8.7 LU (logical unit) state machines

8.7.1 LU state machine overview

The LU (logical unit) state machines manage the queueing of commands, command processing, and task management functions. The LU state machines are as follows:

>   a) LU_TM (task manager) state machine (see 8.7.2);
>   b) LU_DS (device server) state machine (see 8.7.3); and
>   c) LU_CS (command state) state machine (see 8.7.4).

There is one LU_TM state machine and one LU_DS state machine per logical unit. There is one LU_CS state machine for each command placed into the task set by the task manager.

If a state machine consists of multiple states, then the initial state is as indicated in the state machine description.

Figure 43 and figure 44 show the LU state machines.
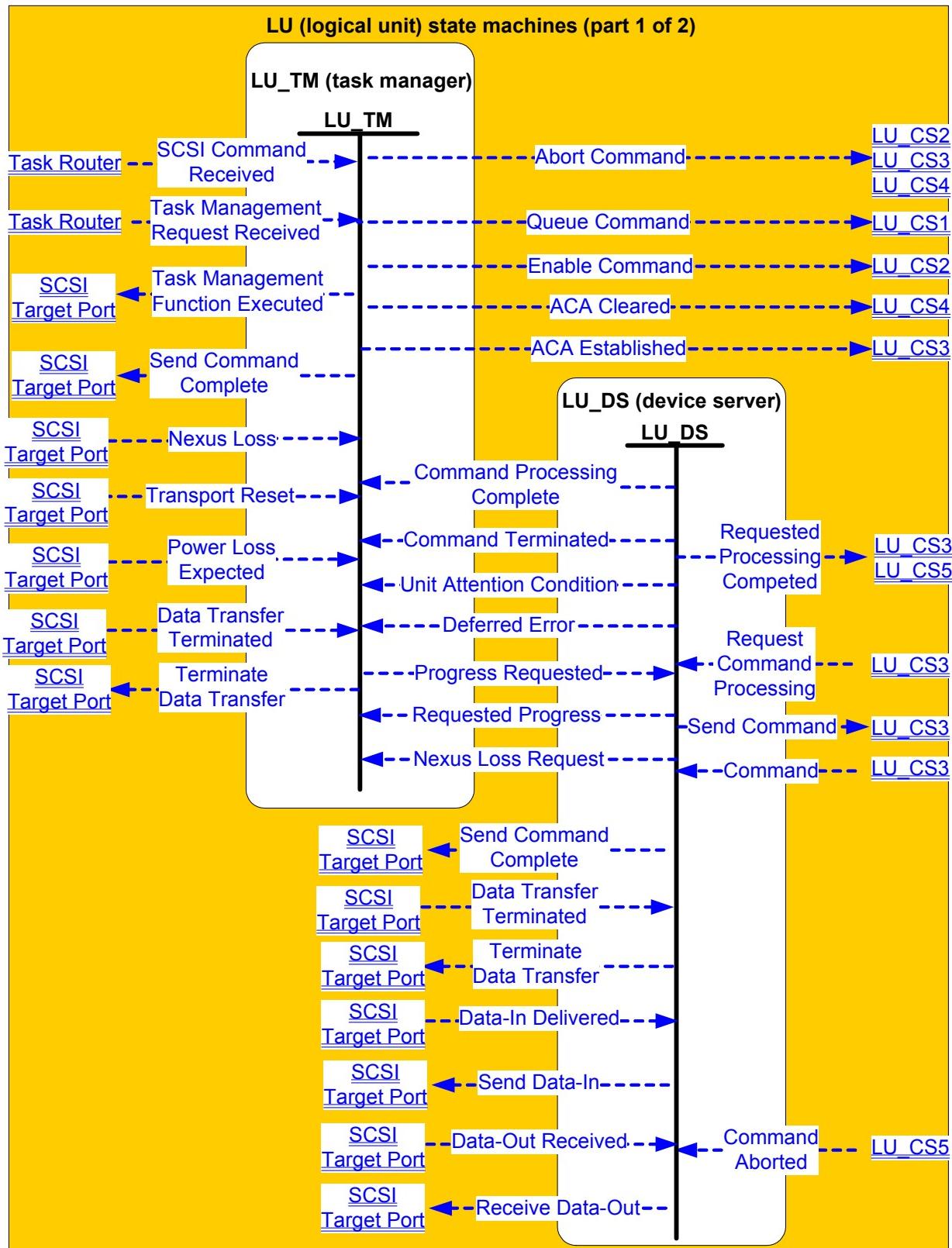
Figure 43 — LU (logical unit) state machines (part 1)
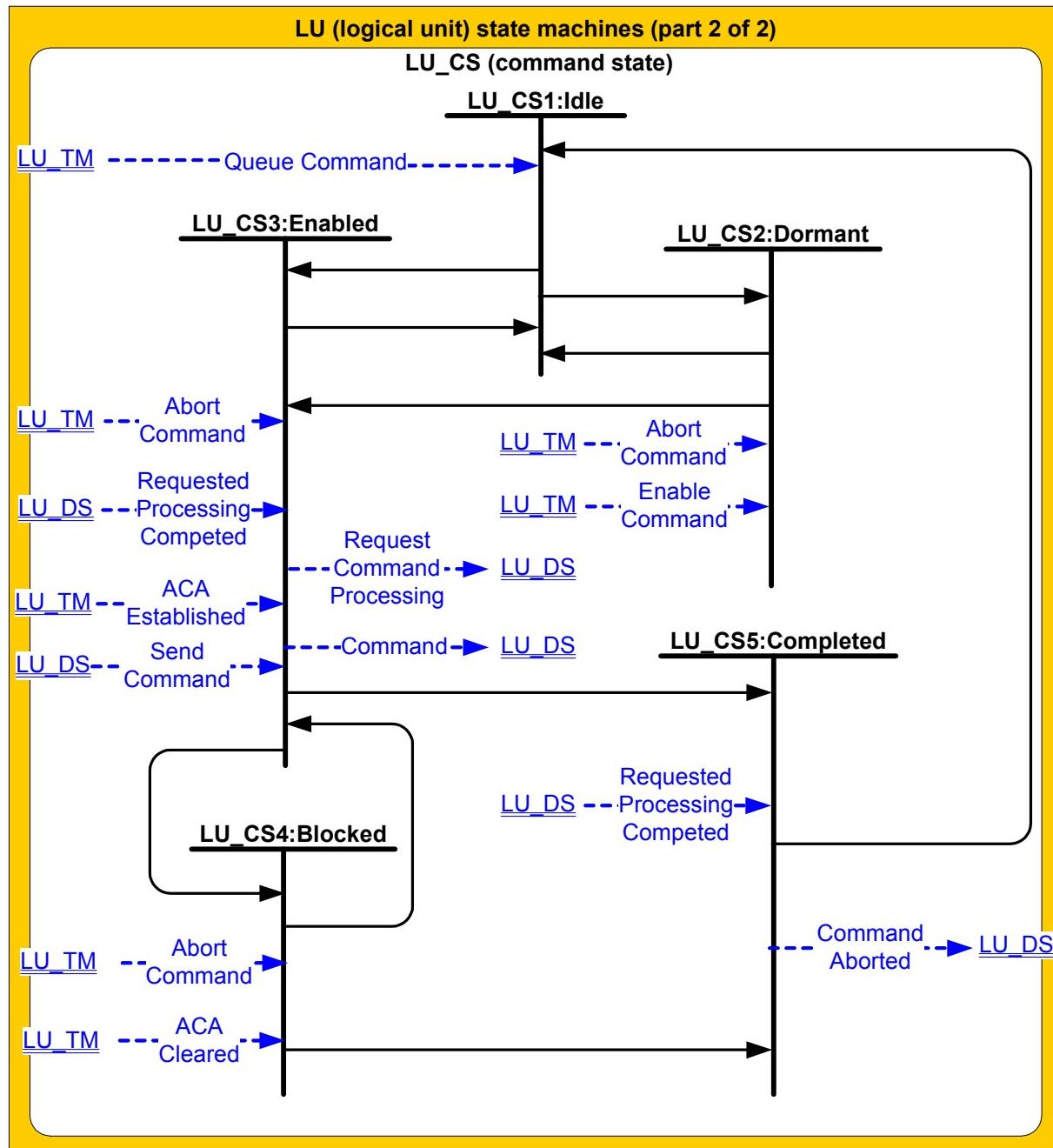
Figure 44 — LU (logical unit) state machines (part 2)

8.7.2 LU_TM (task manager) state machine

8.7.2.1 LU_TM state machine overview

The LU_TM state machine contains the LU_TM state.

The LU_TM state:

   a)   manages commands;
   b)   manages task management functions;

   c)   receives messages from a task router that contain task management functions (see 4.6.24) or commands;

   d)   receives messages from the SCSI target port that indicate a condition occurred as a result of:
       A)   an I_T nexus loss event (see 6.3.4);
       B)   hard reset event (see 6.3.2); or
       C)   power loss expected event (see 6.3.5);

   e)   keeps track of the nexus and command identifier for each command and task management function received;

   f)   keeps track of the association between command identifiers and nexuses;

   g)   keeps track of the task attribute (see 8.4) for each command received;

   h)   receives messages from the LU_DS state machine that indicate:
       A)   unit attention conditions (i.e., Unit Attention Condition message);
       B)   deferred errors (i.e., Deferred Error message); and
       C)   progress of commands being processed (i.e., Requested Progress message);

   i)   sends messages that contain commands as arguments (see 4.6.31) to LU_CS state machines (i.e., Queue Command message);

   j)   sends messages to a LU_CS state machine to control the state of a command (e.g., if the LU_TM state machine determines a specific simple command or ordered command should be processed, then it sends an Enable Command message the LU_CS state machine holding that command causing the command to be transitioned to the LU_CS3:Enabled state);

   k)   sends messages to the LU_DS state to request the progress of a command being processed (i.e., Progress Requested message); and

   l)   sends a message to a SCSI target port to indicate:
       A)   the completion of a task management function (i.e., Task Management Function Executed message); and
       B)   the completion of a command (i.e., Send Command Complete message).

8.7.2.2 LU_TM command processing

If the LU_TM state receives a SCSI Command Received message and the CDB argument contains a command that is defined as an implicit head of queue command (see 8.2), then the LU_TM state may set the Task Attribute argument to Head Of Queue before sending a Queue Command message to a LU_CS state machine.

If the LU_TM state receives a SCSI Command Received message and the TST field in the Control mode page (see SPC-4) is set:

   a)   to 000b, then the LU_TM state shall send a Queue Command message to an LU_CS state machine that does not currently contain a command; or

   b)   to 001b, then the LU_TM state shall send a Queue Command message to an LU_CS state machine that does not currently contain a command in the task set specified by the I_T nexus of the received command.

The Queue Command message shall contain the following arguments:

   a)   I_T_L Nexus;
   b)   Command Identifier;
   c)   CDB;
   d)   Task Attribute;
   e)   CRN, if any;
   f)   Command Priority, if any;
   g)   Duration Expiration Time, if any; and
   h)   First Burst Enabled, if any.

After the LU_TM state sends a Queue Command message to an LU_CS state machine that contains a SIMPLE task attribute or an ORDERED task attribute, the LU_TM state shall monitor all the LU_CS state machines within the task set that contain a command.

If:

   a) no ACA condition has been established (see 5.9); or
   b) an ACA condition has been established, the TST field in the Control mode page (see SPC-4) is set to
       001b, and the command is not from the faulted I_T nexus,

then for a command that has not had an Enable Command message sent (i.e., a not enabled command) and
that has:

   a) a SIMPLE task attribute, the LU_TM state shall send the Enable Command message to the LU_CS
       state machine containing that command if the TST field in the Control mode page (see SPC-4) is set:
       A) to 000b and all commands having:
           a) a HEAD OF QUEUE task attribute have completed; and
           b) an ORDERED task attribute sent to an LU_CS state machine earlier in time than that not
               enabled command have completed;

           or

       B) to 001b and all commands in the task set associated with that command's I_T nexus having:
           a) a HEAD OF QUEUE task attribute have completed; and
           b) an ORDERED task attribute sent to an LU_CS state machine earlier in time than that not
               enabled command have completed;

           or

   b) an ORDERED task attribute, the LU_TM state shall send the Enable Command message to the LU_CS
       state machine containing that command if the TST field in the Control mode page (see SPC-4) is set:
       A) to 000b and all commands:
           a) having a HEAD OF QUEUE task attribute have completed; and
           b) sent to an LU_CS state machine earlier in time than that not enabled command have
               completed;

           or

       B) to 001b and all commands in the task set associated with that command's I_T nexus:
           a) having a HEAD OF QUEUE task attribute have completed; and
           b) sent to an LU_CS state machine earlier in time than that not enabled command have
               completed.

While an ACA condition is established the LU_TM state sends Enable Command messages as described in
8.7.2.5.2.

When the LU_TM state receives a Command Processing Complete message or a Command Terminated
message the LU_TM state shall remove all information for the command specified by the Command Identifier
argument (i.e., releases that location in the task set for another command).

If the LU_TM state receives a SCSI Command Received message and there is no available LU_CS state
machine (i.e., the task set is full), then the LU_TM state shall send a Send Command Complete message to
the SCSI target port with a status of TASK SET FULL as described in 5.3. The Send Command Complete
message shall contain the following arguments:

   a) I_T_L Nexus;
   b) Command Identifier;
   c) Sense Data;
   d) Sense Data Length;
   e) Status;
   f) Service Response; and
   g) Status Qualifier.

If the LU_TM state receives a SCSI Command Received message and there is a condition that causes the logical unit to be busy, then the LU_TM state shall send a Send Command Complete message to the SCSI target port with a status of BUSY as described in 5.3. The Send Command Complete message shall contain the following arguments:

   a) I_T_L Nexus;
   b) Command Identifier;
   c) Sense Data;
   d) Sense Data Length;
   e) Status;
   f) Service Response; and
   g) Status Qualifier.

If the LU_TM state receives a SCSI Command Received message and there:

   a) is a unit attention that has not been cleared (see 5.14) and the received command allows the unit attention to be cleared (see 5.14); or
   b) there is a deferred error that has not been cleared (see SPC-4) and the received command is not an INQUIRY command, REQUEST SENSE command, REPORT LUNS command, or NOTIFY DATA TRANSFER DEVICE command,

then the LU_TM state shall send a Send Command Complete message to the SCSI target port. The Send Command Complete message shall contain the following arguments:

   a) I_T_L Nexus;
   b) Command Identifier;
   c) Sense Data;
   d) Sense Data Length;
   e) Status;
   f) Service Response; and
   g) Status Qualifier.

If the LU_TM state receives a SCSI Command Received message with a Task Attribute argument that is not supported or is not valid (see 5.12) (e.g., set to ACA and there is no ACA condition (see 5.9)), then the LU_TM state shall send a Send Command Complete message to the SCSI target port with arguments set as described in 5.12. The Send Command Complete message shall contain the following arguments:

   a) I_T_L Nexus;
   b) Command Identifier;
   c) Sense Data;
   d) Sense Data Length;
   e) Status;
   f) Service Response; and
   g) Status Qualifier.

If the LU_TM state is required to detect overlapped commands and the LU_TM state detects an overlapped command (see 5.10), then the LU_TM shall process the overlapped command as described in 5.10. For any command that is aborted the LU_TM state shall:

   1) send an Abort Command message to the LU_CS state machine that contains that command;
   2) wait for a Command Processing Complete message for that command; and
   3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument.

8.7.2.3 LU_TM task management function processing

If the LU_TM state receives a Task Management Request Received message, then the LU_TM state shall process the task management function (see table 55) indicated in the Function Identifier argument as described in clause 7.

For any command that is required to be aborted as a result of the task management function processing (see clause 7) the LU_TM state shall:

1) send an Abort Command message to the LU_CS state machine that contains that command;
2) wait for a Command Processing Complete message for that command;
3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument;
4) repeat from step 1) until all specified commands have been aborted; and
5) send a Task Management Function Executed message to the SCSI target port with the following arguments:
   A) Nexus;
   B) Command Identifier; and
   C) Service Response.

To process a Function Identifier argument set to QUERY TASK (see 7.8) for a command specified in the Command Identifier argument that has been sent to a LU_CS state machine, the LU_TM state shall:

1) send a Progress Requested message with a Command Identifier argument set to the Command Identifier specified in the QUERY TASK to the LU_DS state machine;
2) wait for the Requested Progress message;
3) set the Service Response argument and the Additional Response Information argument using the contents of the Requested Progress message Command Progress argument as described in 7.8; and
4) send a Task Management Function Executed message to the SCSI target port with the following arguments:
   A) Nexus;
   B) Command Identifier;
   C) Service Response; and
   D) Additional Response Information.

To process a Function Identifier argument set to QUERY TASK for a command specified in the Command Identifier argument of which the LU_TM state has no knowledge, the LU_TM state shall:

1) set the Service Response argument and the Additional Response Information argument as described in 7.8; and
2) send a Task Management Function Executed message to the SCSI target port with the following arguments:
   A) Nexus;
   B) Command Identifier;
   C) Service Response; and
   D) Additional Response Information.

To process a Function Identifier argument set to QUERY TASK SET (see 7.9) the LU_TM state shall:

1) set the Service Response argument as described in 7.9; and
2) send a Task Management Function Executed message to the SCSI target port with the following arguments:
   A) Nexus;
   B) Command Identifier; and
   C) Service Response.

To process a Function Identifier argument set to QUERY ASYNCHRONOUS EVENT (see 7.10) the LU_TM state shall:

1) set the Service Response argument and the Additional Response Information argument based on information associated with any pending unit attention conditions or pending deferred errors as described in 7.10 and 4.6.25.2.4; and
2) send a Task Management Function Executed message to the SCSI target port with the following arguments:
   A) Nexus;
   B) Command Identifier;
   C) Service Response; and

D) Additional Response Information.

8.7.2.4 LU_TM event processing

If the LU_TM state receives a Nexus Loss message, a Power Loss Expected message, or a Transport Reset message, then the LU_TM state shall process the I_T nexus loss as described in 6.3.4, the power loss expected as described in 6.3.5, and the transport reset as a logical unit reset as described in 6.3.3. For any command that is required to be aborted as a result of the I_T nexus loss processing, power loss expected processing, or logical unit reset processing the LU_TM state shall:

1) send an Abort Command message to the LU_CS state machine that contains that command;
2) wait for a Command Processing Complete message for that command; and
3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument.

If the LU_TM state receives a Nexus Loss Request message, then the LU_TM state shall, for each I_T nexus specified in the REMOVE I_T NEXUS parameter list argument (see SPC-4), process an I_T nexus loss for that I_T nexus as described in 6.3.4. For any command that is required to be aborted as a result of the I_T nexus loss processing the LU_TM state shall:

1) send an Abort Command message to the LU_CS state machine that contains that command;
2) wait for a Command Processing Complete message for that command; and
3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument.

8.7.2.5 LU_TM terminated command processing

8.7.2.5.1 LU_TM ACA not established

If the LU_TM state receives a Command Terminated (Check Condition) message (i.e., no ACA established), then the LU_TM state shall send an Abort Command message to LU_CS state machines based on the contents of the TST field and QERR field in the Control mode page (see SPC-4) as shown in table 61.

Table 61 — Aborting commands already in a task set if an ACA is not established

Field in the Control mode page (see SPC-4)		LU_TM Action
QERR	**TST**	
00b	000b	The LU_TM state takes no additional action (i.e., does not sent any Abort Command messages).
	001b	
01b	000b	For each LU_CS state machine to which the LU_TM state has sent a Queue Command message and the associated command has not completed or terminated, the LU_TM shall: 1) send an Abort Command message; 2) wait for a Command Processing Complete message for the aborted command; and 3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument.
	001b	For each LU_CS state machine in the task set associated with the faulted I_T nexus [a] to which the LU_TM state has sent a Queue Command message and the associated command has not completed or terminated, the LU_TM shall: 1) send an Abort Command message [b]; 2) wait for a Command Processing Complete message for the aborted command; and 3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument.
11b	000b	For each LU_CS state machine in the task set associated with the faulted I_T nexus [a] to which the LU_TM state has sent a Queue Command message and the associated command has not completed or terminated, the LU_TM shall: 1) send an Abort Command message [b]; 2) wait for a Command Processing Complete message for the aborted command; and 3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument.
	001b	

[a] The faulted I_T nexus is the I_T nexus of the Command Terminated (Check Condition) message's I_T_L Nexus argument.
[b] For commands associated with an I_T nexus that is not the faulted I_T nexus the LU_TM state takes no additional action.

8.7.2.5.2 LU_TM ACA established

If the LU_TM state receives a Command Terminated (ACA Condition) message (i.e., ACA established), then the LU_TM state shall send an ACA Established message or an Abort Command message to LU_CS state machines based on the contents of the TST field and QERR field in the Control mode page (see SPC-4) as shown in table 62.

Table 62 — Handling of commands already in a task set if an ACA is established (part 1 of 2)

Field in the Control mode page (see SPC-4)		LU_TM Action
QERR	**TST**	
00b	000b	The LU_TM state shall send an ACA Established message to all LU_CS state machines to which the LU_TM state has sent a Queue Command message and the associated command has not completed or terminated.
	001b	The LU_TM state shall send an ACA Established message to all LU_CS state machines in the task set associated with the faulted I_T nexus [a] [b] to which the LU_TM state has sent a Queue Command message and the associated command has not completed or terminated.
01b	000b	For each LU_CS state machine to which the LU_TM state has sent a Queue Command message and the associated command has not completed or terminated, the LU_TM shall: 1) send an Abort Command message; 2) wait for a Command Processing Complete message for the aborted command; and 3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument.
	001b	For each LU_CS state machine in the task set associated with the faulted I_T nexus [a] to which the LU_TM state has sent a Queue Command message and the associated command has not completed or terminated, the LU_TM shall: 1) send an Abort Command message [b]; 2) wait for a Command Processing Complete message for the aborted command; and 3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument.

[a] The faulted I_T nexus is the I_T nexus of the Command Terminated (ACA Condition) message's I_T_L Nexus argument.
[b] If the LU_TM state receives a SCSI Command Received message with a command associated with an I_T nexus other than the faulted I_T nexus, then that command shall not be affected by the establishment of the ACA condition.

Table 62 — Handling of commands already in a task set if an ACA is established (part 2 of 2)

Field in the Control mode page (see SPC-4)		LU_TM Action
QERR	**TST**	
11b	000b	For each LU_CS state machine associated with the faulted I_T nexus [a] to which the LU_TM state has sent a Queue Command message and the associated command has not completed or terminated, the LU_TM shall: 1) send an Abort Command message; 2) wait for a Command Processing Complete message for the aborted command; and 3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument. The LU_TM state shall send an ACA Established message to all LU_CS state machines not associated with the faulted I_T nexus. [a]
	001b	For each LU_CS state machine in the task set associated with the faulted I_T nexus [a] to which the LU_TM state has sent a Queue Command message and the associated command has not completed or terminated, the LU_TM shall: 1) send an Abort Command message [b]; 2) wait for a Command Processing Complete message for the aborted command; and 3) remove all information for the command specified by the Command Processing Complete message's Command Identifier argument.

[a] The faulted I_T nexus is the I_T nexus of the Command Terminated (ACA Condition) message's I_T_L Nexus argument.

[b] If the LU_TM state receives a SCSI Command Received message with a command associated with an I_T nexus other than the faulted I_T nexus, then that command shall not be affected by the establishment of the ACA condition.

While the ACA condition is in effect, if the LU_TM state receives a SCSI Command Received message from the SCSI target port associated with:

    a) the faulted I_T nexus with a command that is allowed to be processed (see 5.9.3) or a non-faulted I_T nexus with a command that is allowed to be processed (see 5.9.4); and

    b) the TST field in the Control mode page (see SPC-4):
        A) is set to 000b, then the LU_TM state shall send a Queue Command message to an LU_CS state machine that does not currently contain a command; or
        B) is set to 001b, then the LU_TM state shall send a Queue Command message to an LU_CS state machine that does not currently contain a command in the task set specified by the I_T nexus of the received command.

While the ACA condition is in effect, if the LU_TM state receives a SCSI Command Received message from the SCSI target port associated with:

    a) the faulted I_T nexus with a command that is not allowed to enter the task set (see 5.9.3); or
    b) a non-faulted I_T nexus with a command that is not allowed to enter the task set (see 5.9.4),

then the LU_TM state shall send a Send Command Complete message to the SCSI target port with a status described in 5.9. The Send Command Complete message shall contain the following arguments:

    a) I_T_L Nexus;
    b) Command Identifier;
    c) Sense Data, if any;
    d) Sense Data Length, if any;

e) Status;
f) Service Response; and
g) Status Qualifier, if any.

If the ACA condition is cleared as described in 5.9.5 (e.g., LU_TM state receives a Task Management
Request Received message with a Function Identifier argument set to CLEAR ACA), then the LU_TM state
shall send an ACA Cleared message to each LU_CS state machine within the task set associated with the
command that caused the ACA condition.

8.7.3 LU_DS (device server) state machine

8.7.3.1 LU_DS state machine overview

The LU_DS state machine contains the LU_DS state.

The LU_DS state:

a) processes commands;
b) receives messages from the SCSI target port that indicate:
    A) data has been delivered or received; and
    B) a data transfer has been terminated;
c) receives messages from the LU_DS state to request the progress of a command being processed;
d) receives messages from the LU_CS state machine that:
    A) indicate a command is available to be processed (i.e., Request Command Processing message);
    B) contain the command attributes (i.e., Command message); and
    C) indicate a command has been aborted (i.e., Command Aborted message);
e) sends messages to the LU_TM state that indicate:
    A) the processing of a command has completed (i.e., Command Processing Complete message);
    B) the processing of a command has terminated;
    C) the progress in processing a command (i.e., Requested Progress message);
    D) a unit attention condition has occurred (i.e., Unit Attention Condition message); and
    E) a deferred error condition has occurred (i.e., Deferred Error message);
f) sends messages to a LU_CS state machine that:
    A) request the command attributes (Send Command message); or
    B) indicate that the processing of a command is complete (i.e., Command Processing Complete);

    and

g) sends messages to a SCSI target port to:
    A) indicate the completion of a command (i.e., Send Command Complete message);
    B) request data transfers be terminated (i.e., Data Transfer Terminated message); and
    C) request data be transferred (i.e., Send Data-In message and Receive Data-Out message).

A command in the LU_DS state may become a current command and may complete at any time, subject to:

a) the command completion constraints specified in the Control mode page (see SPC-4);
b) the CRN argument, if any, as defined in 5.1; and
c) the Command Priority argument, if any, as defined in 8.5.

Only commands that have been accepted for processing by the LU_DS state shall complete or become a
current command.

Except for the use of resources required to preserve command state, a command shall produce no effects
detectable by the application client before that command is processed by the LU_DS state. Before the LU_DS
state accepts a command for processing, no activity associated with this command shall result in a detectable
change as perceived by an application client.

8.7.3.2 LU_DS command processing

If the LU_DS state receives a Request Command Processing message and there is a condition that results in a reservation conflict (see SPC-4), then the LU_DS state shall send:

  a) a Command Processing Complete message to the LU_TM state with the Command Identifier argument set to the command identifier of the completed command;
  b) a Requested Processing Completed message to the LU_CS state machine associated with the completed command; and
  c) a Send Command Complete message to the SCSI target port with a status of RESERVATION CONFLICT as described in 5.3. The Send Command Complete message shall contain the following arguments:
    A) I_T_L Nexus;
    B) Command Identifier;
    C) Sense Data, if any;
    D) Sense Data Length, if any;
    E) Status;
    F) Service Response; and
    G) Status Qualifier, if any.

If there are multiple commands requesting to be processed, then the LU_DS determines the command to be processed using the following rules:

  a) any command that has a HEAD OF QUEUE task attribute shall be processed before any command that has an ORDERED task attribute or a SIMPLE task attribute; or
  b) if there are no commands that have a HEAD OF QUEUE task attribute, then the processing order for commands that have a SIMPLE task attribute is determined using:
    A) the QUEUE ALGORITHM MODIFIER field in the Control mode page (see SPC-4);
    B) the command priority as defined in 8.5; and
    C) the command duration limit as defined in 8.6.

If the LU_DS state has the resources available to begin processing a command and receives a Request Command Processing message, then the LU_DS state shall:

  1) send a Send Command message to the LU_CS3:Enabled state associated with the received Request Command Processing message;
  2) wait for the Command message;
  3) process the command as described in clause 5 using the arguments received in the Command message; and
  4) if command processing completes with no error, then send:
    A) a Command Processing Complete message to the LU_TM state with the Command Identifier argument set to the command identifier of the completed command;
    B) a Requested Processing Completed message to the LU_CS state machine associated with the completed command; and
    C) a Send Command Complete message to the SCSI target port with the arguments set as defined in 5.4.2.4. The Send Command Complete message shall contain the following arguments:
      a) I_T_L Nexus;
      b) Command Identifier;
      c) Sense Data, if any;
      d) Sense Data Length, if any;
      e) Status;
      f) Service Response; and
      g) Status Qualifier, if any.

If during the processing of a command a Command Aborted message is received for that command, then the LU_DS state shall:

  1) if data is in the process of being transferred from the Data-Out Buffer (see 5.4.3) or into the Data-In Buffer (see 5.4.3), then the LU_DS state shall:

1) send a Terminate Data Transfer message to the SCSI target port with the Nexus argument set to the identifier for the nexus and the Command Identifier argument set to the command identifier associated with the command being terminated; and

2) wait for a Data Transfer Terminated message;

and

2) send:
   A) a Command Processing Complete message to the LU_TM state with the Command Identifier argument set to the command identifier of the aborted command;
   B) a Requested Processing Completed message to the LU_CS state machine associated with the completed command; and
   C) a Send Command Complete message to the SCSI target port with the arguments set as defined in 5.4.2.4. The Send Command Complete message shall contain the following arguments:
      a) I_T_L Nexus;
      b) Command Identifier;
      c) Sense Data, if any;
      d) Sense Data Length, if any;
      e) Status;
      f) Service Response; and
      g) Status Qualifier, if any.

If during the processing of a duration limited command the LU_DS state determines that it is unable to complete the command prior to the duration expiration time (see 8.6), then:

1) if data is in the process of being transferred from the Data-Out Buffer (see 5.4.3) or into the Data-In Buffer (see 5.4.3), then the LU_DS state shall:
   1) send a Terminate Data Transfer message to the SCSI target port with the Nexus argument set to the identifier for the nexus and the Command Identifier argument set to the command identifier associated with the command being terminated; and
   2) wait for a Data Transfer Terminated message;

2) if an ACA condition is established (see 5.9.2), then the LU_DS state shall:
   1) send a Command Terminated (ACA Condition) message to the LU_TM state with a Command Identifier argument set to the command identifier of the terminated command; and
   2) if the TST field in the Control mode page (see SPC-4) is set to:
      a) 000b, then suspend processing commands that have been accepted for processing (i.e., a command received as a result of a Command message before the ACA condition was established); or
      b) 001b, then suspend processing commands for any command associated with the faulted I_T nexus that have been accepted for processing (i.e., a command received as a result of a Command message associated with the faulted I_T nexus before the ACA condition was established);

3) if no ACA condition is established (see 5.8), then the LU_DS state shall send a Command Terminated (Check Condition) message to the LU_TM state with a Command Identifier argument set to the command identifier of the terminated command;

4) the LU_DS state shall send a Requested Processing Completed message to the LU_CS state machine associated with the terminated command; and

5) if:
   A) the LU_DS state machine has not received a Command message associated with the terminated command, then the LU_DS state machine shall send a Send Command Complete message to the SCSI target port that contains the following arguments:
      a) I_T_L Nexus set to the I_T_L Nexus of the terminated command;
      b) Command Identifier set to the command identifier of the terminated command;
      c) Sense Data with the sense key set to ABORTED COMMAND and the additional sense code set to COMMAND TIMEOUT BEFORE PROCESSING;
      d) Sense Data Length; and
      e) Status set to CHECK CONDITION;

      or

B) the LU_DS state machine has received a Command message associated with the terminated command, then the LU_DS state machine shall send a Send Command Complete message to the SCSI target port that contains the following arguments:
   a) I_T_L Nexus set to the I_T_L Nexus of the terminated command;
   b) Command Identifier set to the command identifier of the terminated command;
   c) Sense Data with the sense key set to ABORTED COMMAND and the additional sense code set to COMMAND TIMEOUT DURING PROCESSING or COMMAND TIMEOUT DURING PROCESSING DUE TO ERROR RECOVERY;
   d) Sense Data Length; and
   e) Status set to CHECK CONDITION.

If an error occurs during the processing of a command that results in a CHECK CONDITION status, then:

1) if data is in the process of being transferred from the Data-Out Buffer (see 5.4.3) or into the Data-In Buffer (see 5.4.3), then the LU_DS state shall:
   1) send a Terminate Data Transfer message to the SCSI target port with the Nexus argument set to the identifier for the nexus and the Command Identifier argument set to the command identifier associated with the command being terminated; and
   2) wait for a Data Transfer Terminated message;
2) if an ACA condition is established (see 5.9.2), then the LU_DS state shall:
   1) send a Command Terminated (ACA Condition) message to the LU_TM state with a Command Identifier argument set to the command identifier of the terminated command; and
   2) if the TST field in the Control mode page (see SPC-4) is set to:
      a) 000b, then suspend processing commands that have been accepted for processing (i.e., a command received as a result of a Command message before the ACA condition was established); or
      b) 001b, then suspend processing commands for any command associated with the faulted I_T nexus that have been accepted for processing (i.e., a command received as a result of a Command message associated with the faulted I_T nexus before the ACA condition was established);
3) if no ACA condition is established (see 5.8), then the LU_DS state shall send a Command Terminated (Check Condition) message to the LU_TM state with a Command Identifier argument set to the command identifier of the terminated command; and
4) the LU_DS state shall send:
   A) a Requested Processing Completed message to the LU_CS state machine associated with the terminated command; and
   B) a Send Command Complete message to the SCSI target port with the arguments set as defined in 5.4.2.4. The Send Command Complete message shall contain the following arguments:
      a) I_T_L Nexus;
      b) Command Identifier;
      c) Sense Data, if any;
      d) Sense Data Length, if any;
      e) Status;
      f) Service Response; and
      g) Status Qualifier, if any.

The LU_DS shall start processing commands suspended as the result of an ACA Condition after:

a) a Command Aborted message for a suspended command is received; or
b) a Request Command Processing for a suspended command is received.

If an ACA condition has been established, the LU_DS state is not processing any commands, and a Request Command Processing message is received, then the LU_DS state shall process that command as if no ACA condition has been established.

If the command being processed is a REMOVE I_T NEXUS command (see SPC-4), then the LU_DS state shall send a Nexus Loss Request message with a REMOVE I_T NEXUS parameter list argument to the LU_TM state.

8.7.3.3 LU_DS background processing

If a deferred error has been established (see SPC-4), then the LU_DS state shall send a Deferred Error message to the LU_TM state at least one time.

If a unit attention condition has been established (see 5.14), then the LU_DS state shall send a Unit Attention Condition message to the LU_TM state, at least one time for each unit attention condition.

If the LU_DS state receives a Progress Requested message, then the LU_DS state shall:

1) determine the progress, if any, for the command associated with the Progress Requested message's Command Identifier argument;
2) set the Requested Progress message Command Progress argument as defined in 7.8; and
3) send the Requested Progress message with the Command Progress argument to the LU_TM state.

8.7.4 LU_CS (command state) state machine

8.7.4.1 LU_CS state machine overview

The LU_CS state machine's function is to inform the device server when a command is available to be processed or has been terminated.

The LU_CS state machine consists of the following states:

a) LU_CS1:Idle (see 8.7.4.2) (initial state);
b) LU_CS2:Dorment (see 8.7.4.3);
c) LU_CS3:Enabled (see 8.7.4.4);
d) LU_CS4:Blocked (see 8.7.4.5); and
e) LU_CS5:Completed (see 8.7.4.6).

The LU_CS state machine shall start in the LU_CS1:Idle state.

There is one LU_CS state machine for each command within a task set. The number of commands that may be held within a task set at any time is vendor specific.

8.7.4.2 LU_CS1:Idle state

8.7.4.2.1 LU_CS1:Idle state description

This is the initial state of the LU_CS state machine.

8.7.4.2.2 Transition LU_CS1:Idle to LU_CS2:Dormant

This transition shall occur after receiving a Queue Command message with a Task Attribute argument set to:

a) Simple; or
b) Ordered.

This transition shall include the following arguments:

a) I_T_L Nexus;
b) Command Identifier;
c) CDB;
d) Task Attribute;
e) CRN, if any;
f) Command Priority, if any;
g) Duration Expiration Time, if any; and
h) First Burst Enabled, if any.

8.7.4.2.3 Transition LU_CS1:Idle to LU_CS3:Enabled

This transition shall occur after receiving a Queue Command message with a Task Attribute argument set to:

- a) Head Of Queue; or
- b) ACA Task.

This transition shall include the following arguments:

- a) I_T_L Nexus;
- b) Command Identifier;
- c) CDB;
- d) Task Attribute;
- e) CRN, if any;
- f) Command Priority, if any;
- g) Duration Expiration Time, if any; and
- h) First Burst Enabled, if any.

8.7.4.3 LU_CS2:Dormant state

8.7.4.3.1 LU_CS2:Dormant state description

A command in this state is prevented from being processed by the device server due to the presence of certain other commands in the task set (see 8.4).

Any information the logical unit has or accepts for a command while the command is in this state is required to be held in a condition where it is not available to the command.

8.7.4.3.2 Transition LU_CS2:Dormant to LU_CS3:Enabled

This transition shall occur after receiving:

- a) an Enable Command message.

This transition shall include the following arguments:

- a) I_T_L Nexus;
- b) Command Identifier;
- c) CDB;
- d) Task Attribute;
- e) CRN, if any;
- f) Command Priority, if any;
- g) Duration Expiration Time, if any; and
- h) First Burst Enabled, if any.

8.7.4.3.3 Transition LU_CS2:Dormant to LU_CS1:Idle

This transition shall occur after receiving:

- a) an Abort Command message.

8.7.4.4 LU_CS3:Enabled state

8.7.4.4.1 LU_CS1:Enabled state description

This state:

- a) notifies the LU_DS state machine a command is available to be processed;
- b) delivers a command to the LU_DS state machine;
- c) receives abort command requests; and
- d) receives ACA established notifications.

Upon entry into this state, this state shall:

   a) send a Request Command Processing message to the LU_DS state machine with the following
       arguments:
       A) I_T_L Nexus;
       B) Command Identifier;
       C) CDB;
       D) Task Attribute;
       E) CRN, if any;
       F) Command Priority, if any; and
       G) Duration Expiration Time, if any.

If this state receives a Send Command message, then this state shall:

   a) send a Command message to the LU_DS state machine with the following arguments:
       A) I_T_L Nexus;
       B) Command Identifier;
       C) CDB;
       D) CRN, if any;
       E) Command Priority, if any;
       F) Duration Expiration Time, if any; and
       G) First Burst Enabled, if any.

8.7.4.4.2 Transition LU_CS3:Enabled to LU_CS1:Idle

This transition shall occur after receiving:

   a) a Requested Processing Completed message.

8.7.4.4.3 Transition LU_CS3:Enabled to LU_CS4:Blocked

This transition shall occur after receiving:

   a) an ACA Established message.

This transition shall include the following arguments:

   a) I_T_L Nexus;
   b) Command Identifier;
   c) CDB;
   d) CRN, if any;
   e) Command Priority, if any;
   f) Duration Expiration Time, if any; and
   g) First Burst Enabled, if any.

8.7.4.4.4 Transition LU_CS3:Enabled to LU_CS5:Completed

This transition shall occur after receiving:

   a) an Abort Command message.

This transition shall include the following argument:

   a) Command Identifier.

8.7.4.5 LU_CS4:Blocked state

8.7.4.5.1 LU_CS4:Blocked state description

This state:

   a) receives abort command requests; and

   b)   receives ACA cleared notifications.

Any information the logical unit has or accepts for a command while the command is in this state is required to be held in a condition where it is not available to the command.

8.7.4.5.2 Transition LU_CS4:Blocked to LU_CS3:Enabled

This transition shall occur after receiving:

   a)   an ACA Cleared message.

This transition shall include the following arguments:

   a)   I_T_L Nexus;
   b)   Command Identifier;
   c)   CDB;
   d)   CRN, if any;
   e)   Command Priority, if any;
   f)   Duration Expiration Time, if any; and
   g)   First Burst Enabled, if any.

8.7.4.5.3 Transition LU_CS4:Blocked to LU_CS5:Completed

This transition shall occur after receiving:

   a)   an Abort Command message.

This transition shall include the following argument:

   a)   Command Identifier.

8.7.4.6 LU_CS5:Completed state

8.7.4.6.1 LU_CS5:Completed state description

This state is entered if the task manager has determined a command is to be aborted.

Upon entry into this state, this state shall:

   a)   send a Command Aborted message to the LU_DS state machine with the Command Identifier argument set to the command identifier of the command to be aborted.

8.7.4.6.2 Transition LU_CS5:Completed to LU_CS1:Idle

This transition shall occur after receiving:

   a)   a Requested Processing Completed message.

8.8 Task set management examples

8.8.1 Introduction

Several task set management scenarios are shown in 8.8.2, 8.8.3, and 8.8.4. The examples are valid for configurations with one or more I_T nexuses if the TST field contains 000b (i.e., the interaction among commands in a task set is independent of the I_T nexus on which a command is received). The examples are also valid for a single I_T nexus if the TST field contains 001b (i.e., task set management proceeds independently for each I_T nexus and the events and transitions for the task set associated with one I_T nexus do not affect the task set management for task sets associated with other I_T nexuses). Throughout these examples, the scope of the task set box drawn in each snapshot depends on the setting of the TST field in the Control mode page (see SPC-4).

The figure accompanying each example shows successive snapshots of a task set after various events (e.g., command creation or completion). In all cases, the constraints on command completion order established using Control mode page (see SPC-4) fields other than the TST field (e.g., the QUEUE ALGORITHM MODIFIER field) are not in effect.

A task set is shown as an ordered list or queue of commands with commands having a HEAD OF QUEUE task attribute towards the top of the figure. A new command having a HEAD OF QUEUE task attribute always enters the task set at the head, displacing older commands having a HEAD OF QUEUE task attribute. A command having a SIMPLE task attribute, ORDERED task attribute, or ACA task attribute always enters the task set at the end of the queue.

Command, denoted by rectangles, are numbered in ascending order from oldest to most recent. Fill, shape, and line weight are used to distinguish LU_CS states and task attributes are shown in table 63.

Table 63 — Task attribute and state indications in examples

Task attribute	Box shape	Line weight	State
SIMPLE	Rounded Corners	Thin	LU_CS3:Enabled
ORDERED	Square Corners	Thin	LU_CS2:Dormant
HEAD OF QUEUE	Square Corners	Thick	LU_CS1:Blocked
ACA	Square Corners	Thin Dashed	

The conditions preventing a dormant command from entering LU_CS3:Enabled state, except for ACA conditions, are shown by means of blocking boundaries. Such boundaries appear as horizontal lines with an arrow on both. The commands causing the blocking boundary are described as part of each example. A command is impeded by the blocking boundary if it is between the blocking boundary and the end of the queue. If ACA is not in effect, then a command may enter the LU_CS3:Enabled state after all intervening barriers have been removed.

8.8.2 Commands having the HEAD OF QUEUE task attribute

Figure 45 shows task set conditions when several commands having a HEAD OF QUEUE task attribute are processed.
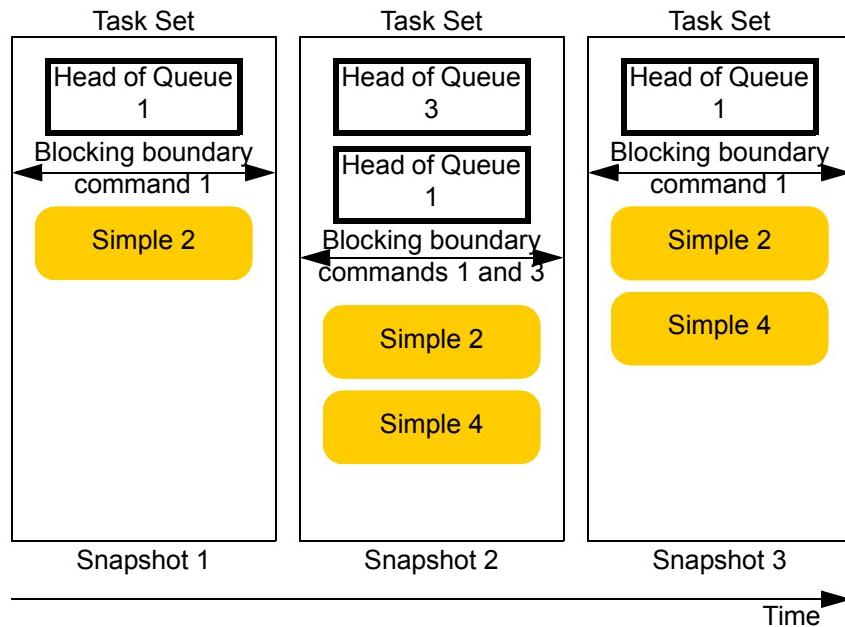
Figure 45 — Commands having the HEAD OF QUEUE task attribute and blocking boundaries (example 1)

In snapshot 1 the task set contains one command having a HEAD OF QUEUE task attribute and one command having a SIMPLE task attribute. As shown by the blocking boundary, command having a SIMPLE task attribute 2 is in the LU_CS2:Dormant state because of the command having a HEAD OF QUEUE task attribute. Snapshot 2 shows the task set after the command having a HEAD OF QUEUE task attribute 3 and the command having a SIMPLE task attribute 4 are created. The new command having a HEAD OF QUEUE task attribute is placed at the front of the queue in the LU_CS3:Enabled state, displacing command 1. Snapshot 3 shows the task set after command 3 completes. Since the conditions indicated by the command 1 blocking boundary are still in effect, command 2 and command 4 remain in the LU_CS2:Dormant state.

Figure 46 is the same as the previous example, except that command 1 completes instead of command 3.
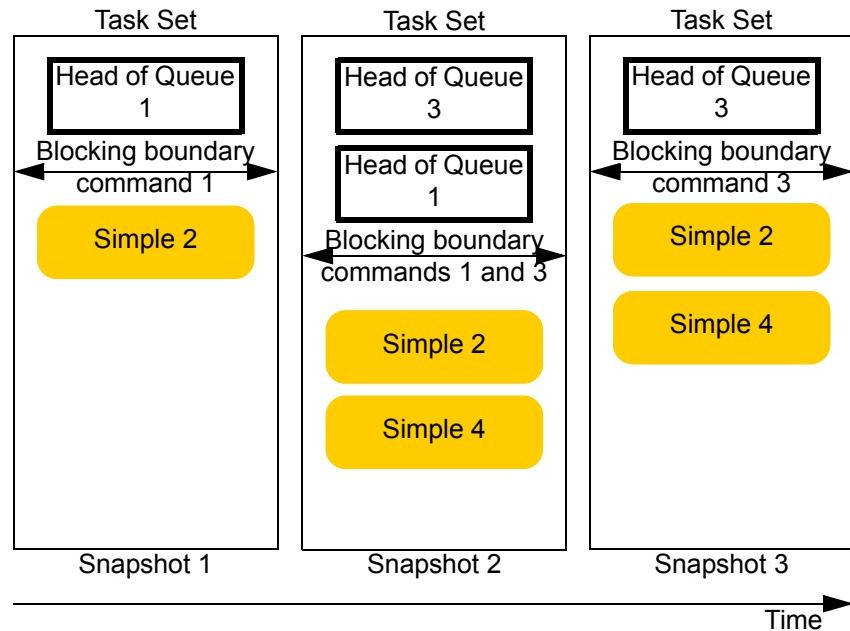
**Figure 46 — Commands having the HEAD OF QUEUE task attribute and blocking boundaries (example 2)**

Because the blocking boundary remains in place for a command having a HEAD OF QUEUE task attribute, both the commands having a SIMPLE task attribute remain in the LU_CS2:Dormant state in snapshot 3. The blocking boundary is not removed until all commands having a HEAD OF QUEUE task attribute complete.

8.8.3 Commands having the ORDERED task attribute

An example of commands having ORDERED task attributes and commands having SIMPLE task attributes interaction is shown in figure 47.
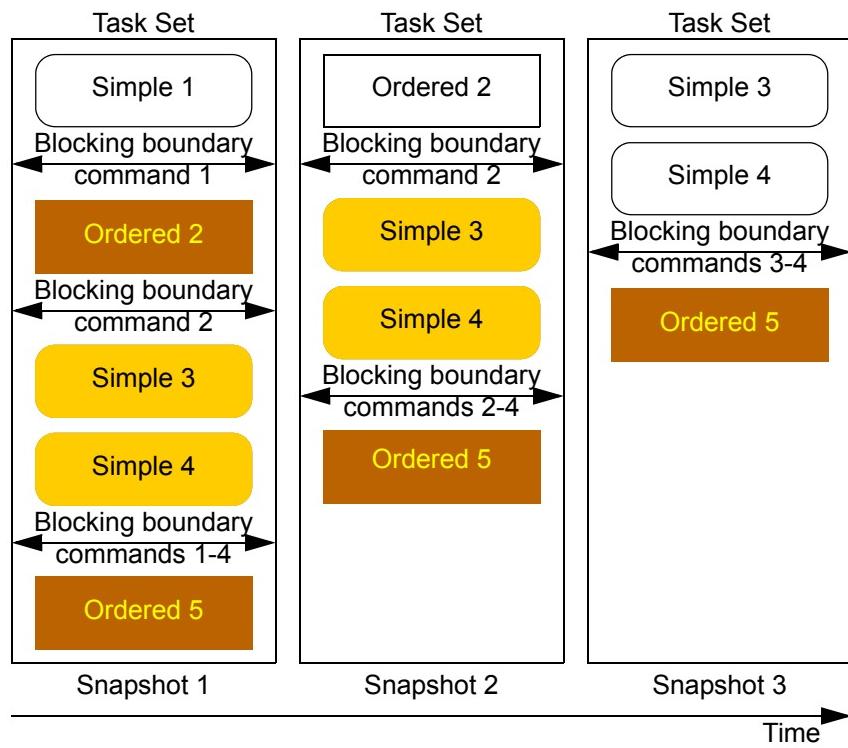
Figure 47 — Commands having ORDERED task attributes and blocking boundaries

The state of dormant command 2 to command 5 is determined by the requirements shown in table 64.

Table 64 — Dormant command blocking boundary requirements

Command	Reason for blocking boundary
2	A command having an ORDERED task attribute is not allowed to enter the LU_CS3:Enabled state until all commands having a HEAD OF QUEUE task attribute and all older commands have completed.
5	
3	A command having a SIMPLE task attribute is not allowed to enter the LU_CS3:Enabled state until all commands having a HEAD OF QUEUE task attribute and all older ordered commands have completed.
4	

The table 64 constraints are shown by the blocking boundaries in snapshot 1.

In snapshot 2, the completion of command 1 allows the command having an ORDERED task attribute 2 to enter the LU_CS3:Enabled state. Since the initial constraints on command 3, command 4 and command 5 are still in effect, these commands are required to remain in the LU_CS2:Dormant state. As shown in snapshot 3, the completion of command 2 triggers two state transitions, with command 3 and command 4 transitioning to the LU_CS3:Enabled state. Command 5 is required to remain in the LU_CS2:Dormant state until command 3 and command 4 complete.

8.8.4 Commands having the ACA task attribute

Figure 48 shows the effects of an ACA condition on the task set. This example assumes the QERR field contains 00b in the Control mode page (see SPC-4). Consequently, clearing an ACA condition does not cause commands to be aborted.
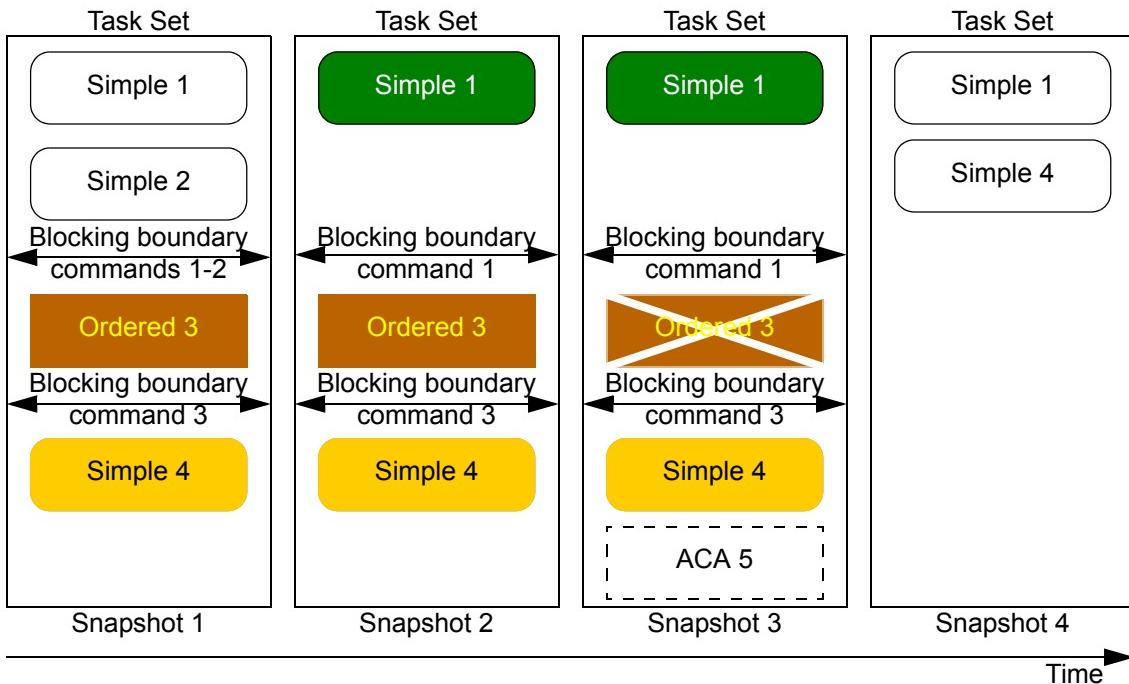
Figure 48 — Commands having ACA task attributes example

The completion of command 2 with CHECK CONDITION status causes command 1 to enter the LU_CS4:Blocked state shown in snapshot 2. In snapshot 3, command having an ORDERED task attribute 3 is aborted using the ABORT TASK task management function and command having an ACA task attribute 5 is created to perform additional handling for the exception. Once the ACA condition is cleared (i.e., snapshot 4), command having a SIMPLE task attribute 1 is allowed to transition back to the LU_CS3:Enabled command state. Since there are no commands having a HEAD OF QUEUE task attribute or older commands having an ORDERED task attribute, command 4 also transitions to the LU_CS3:Enabled command state.

Annex A
(informative)

Identifiers and names for objects

A.1 Identifiers and names overview

This annex summarizes identifiers and names.

The following SCSI architecture model objects have identifiers and names summarized in this annex:

    a)   SCSI initiator port;
    b)   SCSI target port;
    c)   logical unit; and
    d)   SCSI device.

A.2 Identifiers and names

This standard defines the identifier attributes and name attributes listed in A.1. The size requirements placed on identifier attributes by this standard are as shown in table A.1. This standard places no requirements on the sizes of name attributes as shown in table A.2.

Table A.1 — Identifier attribute size and support requirements

Attribute	Identifier	
	Size	**Support requirements**
Initiator port identifier	not specified	optional
Target port identifier	not specified	optional
LUN	2 bytes or 8 bytes [a]	mandatory [b] optional (subsidiary logical unit)

[a] Defined in the SCSI transport protocol.
[b] Mandatory for administrative logical units, hierarchical logical units, management logical units, and well known logical units.

Table A.2 — Name attribute size and support requirements

Attribute	Name	
	Size	**Support requirements [a]**
SCSI device name	not specified	optional
Initiator port name	not specified	optional
Target port name	not specified	optional
Logical unit name	not specified	mandatory (logical unit) none (well known logical unit)

[a] As defined in this standard.

Each SCSI transport protocol defines the size and format of identifier attributes and name attributes.

See table A.3 for a list of the sizes for identifier attributes for each SCSI transport protocol.

Table A.3 — Identifier attribute size for each SCSI transport protocol

Attribute	Size		
SCSI transport protocol	**Initiator port identifier**	**Target port identifier**	**LUN**
sADT (see ADT-2)	none	none	2 bytes
iADT (see ADT-2)	16 bytes	16 bytes	2 bytes
FCP (see FCP-4)	3 bytes	3 bytes	8 bytes
iSCSI (see iSCSI)	241 bytes [a]	233 bytes [a]	8 bytes
SBP (see SBP-3)	2 bytes	11 bytes	2 bytes
SOP (see SOP)	2 bytes	2 bytes	8 bytes
SAS SSP (see SPL-3)	8 bytes	8 bytes	8 bytes
SRP (see SRP)	16 bytes	16 bytes	8 bytes
UAS (see UAS)	none	15 bits	8 bytes
Key: none = objects using this SCSI transport protocol do not have this attribute			
[a] Maximum size, including the terminating null character byte.			

See table A.4 for a list of the format of the identifier attributes for each SCSI transport protocol.

Table A.4 — Identifier attribute format for each SCSI transport protocol

Attribute	Format		
SCSI transport protocol	**Initiator port identifier**	**Target port identifier**	**LUN**
sADT (see ADT-2)	none	none	as specified in this standard (2 byte version only see 4.7)
iADT (see ADT-2)	IP address	IP address	as specified in this standard (2 byte version only see 4.7)
FCP (see FCP-4)	Fibre Channel address identifier	Fibre Channel address identifier	as specified in this standard (see 4.7)
iSCSI (see iSCSI)	iSCSI name [b] || ',i,' hex prefix [c] || Initiator Session Identifier [d]	iSCSI name [b] || ',t,' hex prefix [c] || Target Portal Group Tag [e]	as specified in this standard (see 4.7)
SBP (see SBP-3)	binary value	EUI-64 || Discovery ID [f]	as specified in this standard (2 byte version only see 4.7)
SOP (see SOP)	PCI Express routing ID	PCI Express routing ID	as specified in this standard (see 4.7)
SAS SSP (see SPL-3)	NAA IEEE Registered format or NAA Locally Assigned format	NAA IEEE Registered format or NAA Locally Assigned format	as specified in this standard (see 4.7)
SRP (see SRP)	EUI-64 || 8 byte extension [g]	EUI-64 || 8 byte extension [g]	as specified in this standard (see 4.7)
UAS (see UAS)	none	USB device address || USB interface number	as specified in this standard (see 4.7)

Key:
|| = "concatenated with"
none = objects using this SCSI transport protocol do not have this attribute

[a] iSCSI identifiers are concatenated strings containing no null characters except after the last string in the concatenation.
[b] The iSCSI name portion of the string is a world wide unique UTF-8 string no more than 223 bytes long, not including null character termination.
[c] The hex prefix is a UTF-8 string containing '0x' or '0X'.
[d] The Initiator Session Identifier (ISID) portion of the string is a UTF-8 encoded hexadecimal representation of a six byte binary value. This portion of the string contains no more than 12 bytes, not including null character termination if any, and contains Arabic numerals 0 to 9 and/or lower-case or upper-case English letters A to F.
[e] The Target Portal Group Tag (TPGT) portion of the string is a UTF-8 encoded hexadecimal representation of a two byte binary value. This portion of the string contains no more than 4 bytes, not including null character termination if any, and contains Arabic numerals 0 to 9 and/or lower-case or upper-case English letters A to F.
[f] See ISO/IEC 13213:1994 for more information on the Discovery ID.
[g] Required to be world wide unique and recommend to be EUI-64 concatenated with an eight byte extension.

See table A.5 for a list of the size of the name attributes for each SCSI transport protocol.

Table A.5 — Name attribute size for each SCSI transport protocol

Attribute	Size [a]			
SCSI transport protocol	**SCSI device name**	**Initiator port name**	**Target port name**	**Logical unit name**
sADT (see ADT-2)	not specified	none	none	as specified in the Device Identification VPD page (see SPC-4)
iADT (see ADT-2)	not specified	not specified	not specified	
FCP (see FCP-4)	8 bytes	8 bytes	8 bytes	
iSCSI [b] (see iSCSI)	224 bytes	241 bytes	233 bytes	
SBP (see SBP-3)	not specified	8 bytes	11 bytes	
SOP (see SOP)	8 bytes	8 bytes	8 bytes	
SAS SSP (see SPL-3)	8 bytes	none	none	
SRP (see SRP)	not specified	16 bytes	16 bytes	
UAS (see UAS)	not specified	none	none	8 bytes or 16 bytes [c]

Key:
not specified = objects using this SCSI transport protocol may or may not have this attribute and the SCSI transport protocol standard does not specify a size for this attribute
none = objects using this SCSI transport protocol do not have this attribute

[a] Any SCSI transport protocol may support the SCSI name string format (see SPC-4), resulting in names with the sizes shown in the iSCSI column.
[b] Maximum size, including the terminating null character byte.
[c] if the NAA is 2h (i.e., IEEE Extended), 3h (i.e., Locally Assigned), or 5h (i.e., IEEE Registered), then the size is eight bytes. If the NAA is 6h (i.e., IEEE Registered Extended), then the size is 16 bytes.

See table A.6 for a list of the format of the name attributes for each SCSI transport protocol.

Table A.6 — Name attribute format for each SCSI transport protocol

Attribute	Format [a]			
SCSI transport protocol	**SCSI device name**	**Initiator port name**	**Target port name**	**Logical unit name**
sADT (see ADT-2)	not specified	none	none	as specified in the Device Identification VPD page name (see SPC-4)
iADT (see ADT-2)	not specified	not specified	not specified	
FCP (see FCP-4)	NAA IEEE Registered format	Fibre Channel Name_Identifier	Fibre Channel Name_Identifier	
iSCSI (see iSCSI)	SCSI name string format	iSCSI name [c] || 'i,' hex prefix [d] || Initiator Session Identifier [e]	iSCSI name [c] || 't,' hex prefix [d] || Target Portal Group Tag [f]	
SBP (see SBP-3)	not specified	EUI-64	EUI-64 || Discovery ID [g]	
SOP (see SOP)	NAA IEEE Registered format or NAA Locally Assigned format	NAA IEEE Registered format or NAA Locally Assigned format	NAA IEEE Registered format or NAA Locally Assigned format	
SAS SSP (see SPL-3)	NAA IEEE Registered format or NAA Locally Assigned format	none	none	
SRP (see SRP)	not specified	EUI-64 || 8 byte extension [h]	EUI-64 || 8 byte extension [h]	
UAS (see UAS)	not specified	none	none	NAA format

Key:
|| = "concatenated with"
not specified = objects using this SCSI transport protocol may or may not have this attribute and the SCSI transport protocol standard does not specify a size for this attribute
none = objects using this SCSI transport protocol do not have this attribute

[a] In addition to the name formats shown in this table, any SCSI transport protocol may support the SCSI name string format (see SPC-4).

[b] iSCSI identifiers are concatenated strings containing no null characters except after the last string in the concatenation.

[c] The iSCSI name portion of the string is a world wide unique UTF-8 string no more than 223 bytes long, not including null character termination.

[d] The hex prefix is a UTF-8 string containing '0x' or '0X'.

[e] The Initiator Session Identifier (ISID) portion of the string is a UTF-8 encoded hexadecimal representation of a six byte binary value. This portion of the string contains no more than 12 bytes, not including null character termination if any, and contains Arabic numerals 0 to 9 and/or lower-case or upper-case English letters A to F.

[f] The Target Portal Group Tag (TPGT) portion of the string is a UTF-8 encoded hexadecimal representation of a two byte binary value. This portion of the string contains no more than four bytes, not including null character termination if any, and contains Arabic numerals 0 to 9 and/or lower-case or upper-case English letters A to F.

[g] See ISO/IEC 13213:1994 for more information on the Discovery ID.

[h] Required to be world wide unique and recommended to be EUI-64 concatenated with an eight byte extension.

Annex B
(informative)

SCSI Initiator Port attributes and SCSI Target Port attributes supported by SCSI transport protocols

Table B.1 and table B.2 list the values of the SCSI Initiator Port attributes that a SCSI initiator port using each different SCSI transport protocol is able to return, and the values of the SCSI Target Port attributes that a SCSI target port using that SCSI transport protocol is able to return.

Table B.1 — SCSI Initiator Port attributes and SCSI Target Port attributes that are supported by ADT-2, FCP-4, and iSCSI SCSI transport protocols (part 1 of 2)

Attribute	ADT (see ADT-2)	FCP (see FCP-4)	iSCSI (see iSCSI)
LUN size (in bits)	16	64	64
Maximum CDB length [a] (in bytes)	65 523	268	65 550
Command identifier name	Exchange ID	see [b]	Initiator Task Tag
Command identifier size (in bits)	3	see [c]	32
Command identifier scope	I_T nexus	I_T nexus	I_T nexus
Command identifier scope shared with	all exchanges	all exchanges	all iSCSI tasks
Task Attributes supported	SIMPLE, HEAD OF QUEUE, ORDERED, and ACA		
Maximum Data-In Buffer Size (in bytes)	FFFF FFFFh		
Maximum Data-Out Buffer Size (in bytes)	FFFF FFFFh		
Maximum CRN [d]	zero	FFh	zero
Command Priority supported	no	yes	yes [g]
Maximum Sense Data Length [e] (in bytes)	FFFFh	FFFF FFFFh	FFFFh
Status Qualifier supported	no	yes	yes [g]
Additional Response Information supported	no	yes	yes [g]
Bidirectional Commands supported	yes		

[a] SPC-4 defines the maximum length of a CDB as being 260 bytes.
[b] OX_ID if task retry identification is not active, or OX ID and task retry identifier if task retry identification is active.
[c] 16 bits if task retry identification is not active or 48 bits if task retry identification is active.
[d] Maximum CRN of zero indicates that CRN is not supported.
[e] SPC-4 defines the maximum length of sense data as being 252 bytes.
[f] The task management function name is the name of the procedure call, not an argument to a procedure call.
[g] Only supported by RFC7144 compliant implementations.
[h] The QUERY TASK, QUERY TASK SET, I_T_NEXUS RESET, and QUERY ASYNCHRONOUS EVENT are only supported by RFC7144 compliant implementations.

Table B.1 — SCSI Initiator Port attributes and SCSI Target Port attributes that are supported by ADT-2, FCP-4, and iSCSI SCSI transport protocols (part 2 of 2)

Attribute	ADT (see ADT-2)	FCP (see FCP-4)	iSCSI (see iSCSI)
Task Management Functions supported [f]	ABORT TASK, ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, CLEAR ACA, QUERY TASK	ABORT TASK, ABORT TASK SET, CLEAR TASK SET, CLEAR ACA, LOGICAL UNIT RESET, QUERY TASK, QUERY TASK SET, QUERY ASYNCHRONOUS EVENT	all [h]
ABORT TASK scope	I_T_L nexus	I_T nexus	I_T_L nexus
QUERY TASK scope	I_T_L nexus	I_T nexus	I_T_L nexus

[a] SPC-4 defines the maximum length of a CDB as being 260 bytes.

[b] OX_ID if task retry identification is not active, or OX ID and task retry identifier if task retry identification is active.

[c] 16 bits if task retry identification is not active or 48 bits if task retry identification is active.

[d] Maximum CRN of zero indicates that CRN is not supported.

[e] SPC-4 defines the maximum length of sense data as being 252 bytes.

[f] The task management function name is the name of the procedure call, not an argument to a procedure call.

[g] Only supported by RFC7144 compliant implementations.

[h] The QUERY TASK, QUERY TASK SET, I_T_NEXUS RESET, and QUERY ASYNCHRONOUS EVENT are only supported by RFC7144 compliant implementations.

Table B.2 — SCSI Initiator Port attributes and SCSI Target Port attributes that are supported by SOP, SAS SSP, SRP, and UAS SCSI transport protocols

Attribute	SOP over PQI (see SOP)	SAS SSP (SPL-3)	SRP (see SRP)	UAS (see UAS)
LUN size (in bits)	64	64	64	64
Maximum CDB length [a] (in bytes)	268	268	268	268
Command identifier name	request identifier and outbound queue identifier	initiator port transfer tag	tag	tag
Command identifier size (in bits)	32	16	64	16
Command identifier scope	I_T nexus	I_T nexus	I_T nexus	I_T nexus
Command identifier scope shared with	task management functions, SOP management functions	task management functions	all SRP requests	task management functions
Task Attributes supported	SIMPLE, HEAD OF QUEUE, ORDERED, and ACA			
Maximum Data-In Buffer Size (in bytes)	FFFF FFFFh			none
Maximum Data-Out Buffer Size (in bytes)	FFFF FFFFh			none
Maximum CRN [b]	zero			
Command Priority supported	yes	yes	no	yes
Maximum Sense Data Length [c] (in bytes)	FFFFh	1 000	FFFF FFFFh	252
Status Qualifier supported	yes	yes	no	yes
Additional Response Information supported	yes	yes	no	yes
Bidirectional Commands supported	yes			
Task Management Functions supported [d]	all	all	ABORT TASK, ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, CLEAR ACA	all
ABORT TASK scope	I_T_L nexus	I_T_L nexus	I_T_L nexus	I_T_L nexus
QUERY TASK scope	I_T_L nexus	I_T_L nexus	I_T_L nexus	I_T_L nexus

[a] SPC-4 defines the maximum length of a CDB as being 260 bytes.
[b] Maximum CRN of zero indicates that CRN is not supported.
[c] SPC-4 defines the maximum length of sense data as being 252 bytes.
[d] The task management function name is the name of the procedure call, not an argument to a procedure call.

Annex C

(informative)

Terminology mapping

C.1 Terminology mapping to SAM-3

The introduction of a UML model into this standard resulted in changes in terminology between this standard and SAM-3 (see table C.1).

Table C.1 — Terminology mapping to SAM-3

Term used in this standard	Term used in SAM-3
command identifier	task tag
command	task

C.2 Terminology mapping to SAM-4

The removal of the I_T_L_Q nexus term and changes to the queueing state machine from this standard resulted in changes in terminology between this standard and SAM-4 (see table C.2).

Table C.2 — Terminology mapping to SAM-4

Term used in this standard	Term used in SAM-4
I_T_L nexus and command identifier	I_T_L_Q nexus
enabled command	enabled command state
dormant command	dormant command state
blocked command	blocked command state

Annex D
(informative)

SCSI transport protocol acronyms

EUI-64 (Extended Unique Identifier, a 64-bit globally unique identifier):
The IEEE maintains a tutorial describing EUI-64 at http://standards.ieee.org/regauth/oui/tutorials/EUI64.html.

NAA:
Name Address Authority (see SPC-4).

SAS SSP:
SAS Serial SCSI Protocol-3 (see SPL-3).

UTF-8:
See ISO/IEC 10646-1:2000.

Bibliography

ISO/IEC 13213:1994, *Information technology - Microprocessor systems - Control and Status Registers Architecture for microcomputer buses [ANSI/IEEE 1212, 1994 Edition]. See http://www.iso.org/*.

ISO/IEC 14776-154, *Serial Attached SCSI - 2 (SAS-3)* (under consideration)

ISO/IEC 14776-192, *Automation/Drive Interface - Transport Protocol - 2 (ADT-2)*

ISO/IEC 14776-224 *Fibre Channel Protocol for SCSI - 4 (FCP-4)*

ISO/IEC 14776-232, *Serial Bus Protocol - 3 (SBP-3)*

ISO/IEC 14776-241, *SCSI RDMA Protocol (SRP)*

ISO/IEC 14776-342, *SCSI Controller Commends - 2 (SCC-2)*

ISO/IEC 14776-412, *Information technology - Small computer system interface (SCSI) - Part 412: Architecture model-2 (SAM-2)*.

ISO/IEC 14776-413, *Information technology - Small computer system interface (SCSI) - Part 412: Architecture model-3 (SAM-3)*.

ISO/IEC 14776-414, *Information technology - Small computer system interface (SCSI) - Part 412: Architecture model-4 (SAM-4)*.

ISO/IEC 10646-1:2000, *Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane. See http://www.iso.org/*.

ISO/IEC 14776-263, *Information technology - Small computer system interface (SCSI) - Part 261: SAS Protocol Layer - 3 (SPL-3)* (under consideration)

T10/BSR INCITS 489, *SCSI over PCIe (SOP)* (under national consideration).

RFC7143, *iSCSI Protocol (Consolidated)*

> NOTE 16 - Copies of this IETF standards may be obtained through the Internet Engineering Task Force (IETF) at http://www.ietf.org/rfc/rfc7143.txt.

RFC7144, *Internet Small Computer Systems Interface (iSCSI) SCSI Features Update*

> NOTE 17 - A copy of this IETF standard may be obtained through the Internet Engineering Task Force (IETF) at http://www.ietf.org/rfc/rfc7144.txt.

ISO/IEC 19501, *Unified Modeling Language Specification Version 1.4.2*

> NOTE 18 - For more information on the UML specification, contact the Object Modeling Group at http://www.omg.org.